VISIT…

Bernd Jähne

with CD-ROM · with CD-ROM ·

Digital Image Processing

5th revised and extended edition

Springer

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Tokyo

Bernd Jähne

Digital Image Processing

5th revised and extended edition
with 248 figures and CD–ROM

Springer

Prof. Dr. Bernd Jähne
University of Heidelberg
Interdisciplinary Center for Scientific Computing
Im Neuenheimer Feld 368
69120 Heidelberg
Germany
e-mail: Bernd. Jaehne@iwr.uni-heidelberg.de

ISBN 3-540-67754-2 Springer-Verlag Berlin Heidelberg New York

Library of Congress Cataloging-in-Publication-Data

Jähne, Bernd:
Digital image processing : with CD-ROM / Bernd Jähne. - Berlin ; Heidelberg ; New York ; Barcelona ;
Hong Kong ; London ; Milan ; Paris ; Tokyo : Springer, 2002
(Engineering online library)
ISBN 3-540-67754-2

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitations, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

© Springer-Verlag Berlin Heidelberg 2002
Printed in Germany

The use of general descriptive names, registered names trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: Data delivered by author
Cover Design: Struve & Partner, Heidelberg
Printed on acid free paper SPIN: 10774465 62/3020/M – 5 4 3 2 1 0

Preface to the Fifth Edition

As the fourth edition, the fifth edition is completely revised and extended. The whole text of the book is now arranged in 20 instead of 16 chapters. About one third of text is marked as advanced material by a smaller typeface and the † symbol in the headlines. In this way, you will find a quick and systematic way through the basic material and you can extend your studies later to special topics of interest.

The most notable extensions include a detailed discussion on random variables and fields (Chapter 3), 3-D imaging techniques (Chapter 8) and an approach to regularized parameter estimation unifying techniques including inverse problems, adaptive filter techniques such as anisotropic diffusion, and variational approaches for optimal solutions in image restoration, tomographic reconstruction, segmentation, and motion determination (Chapter 17). You will find also many other improvements and additions throughout the whole book. Each chapter now closes with a section "Further Reading" that guides the interested reader to further references.

There are also two new appendices. Appendix A gives a quick access to a collection of often used reference material and Appendix B details the notation used throughout the book.

The complete text of the book is now available on the accompanying CD-ROM. It is hyperlinked so that it can be used in a very flexible way. You can jump from the table of contents to the corresponding section, from citations to the bibliography, from the index to the corresponding page, and to any other cross-references.

The CD-ROM contains a full set of exercises to all topics covered by this book. Using the image processing software heurisko that is included on the CD-ROM you can apply in practice what you have learnt theoretically. A large collection of images, image sequences, and volumetric images is available for practical exercises. The exercises and image material are frequently updated. The newest version is available on the Internet at the homepage of the author (http://klimt.iwr.uni-heidelberg.de).

I would like to thank all individuals and organizations who have contributed visual material for this book. The corresponding acknowledgements can be found where the material is used. I would also like to

express my sincere thanks to the staff of Springer-Verlag for their constant interest in this book and their professional advice. Special thanks are due to my friends at AEON Verlag & Studio, Hanau, Germany. Without their dedication and professional knowledge it would not have been possible to produce this book and the accompanying CD-ROM.

Finally, I welcome any constructive input from you, the reader. I am grateful for comments on improvements or additions and for hints on errors, omissions, or typing errors, which — despite all the care taken — may have slipped attention.

Heidelberg, November 2001 Bernd Jähne

From the preface of the fourth edition

In a fast developing area such as digital image processing a book that appeared in its first edition in 1991 required a complete revision just six years later. But what has not changed is the proven concept, offering a systematic approach to digital image processing with the aid of concepts and general principles also used in other areas of natural science. In this way, a reader with a general background in natural science or an engineering discipline is given fast access to the complex subject of image processing. The book covers the basics of image processing. Selected areas are treated in detail in order to introduce the reader both to the way of thinking in digital image processing and to some current research topics. Whenever possible, examples and image material are used to illustrate basic concepts. It is assumed that the reader is familiar with elementary matrix algebra and the Fourier transform.

The new edition contains four parts. Part 1 summarizes the basics required for understanding image processing. Thus there is no longer a mathematical appendix as in the previous editions. Part 2 on image acquisition and preprocessing has been extended by a detailed discussion of image formation. Motion analysis has been integrated into Part 3 as one component of feature extraction. Object detection, object form analysis, and object classification are put together in Part 4 on image analysis.

Generally, this book is not restricted to 2-D image processing. Wherever possible, the subjects are treated in such a manner that they are also valid for higher-dimensional image data (volumetric images, image sequences). Likewise, color images are considered as a special case of multichannel images.

Heidelberg, May 1997 Bernd Jähne

From the preface of the first edition

Digital image processing is a fascinating subject in several aspects. Human beings perceive most of the information about their environment through their visual sense. While for a long time images could only be captured by photography, we are now at the edge of another technological revolution which allows image data to be captured, manipulated, and evaluated electronically with computers. With breathtaking pace, computers are becoming more powerful

and at the same time less expensive, so that widespread applications for digital image processing emerge. In this way, image processing is becoming a tremendous tool for analyzing image data in all areas of natural science. For more and more scientists digital image processing will be the key to study complex scientific problems they could not have dreamed of tackling only a few years ago. A door is opening for new interdisciplinary cooperation merging computer science with the corresponding research areas.

Many students, engineers, and researchers in all natural sciences are faced with the problem of needing to know more about digital image processing. This book is written to meet this need. The author — himself educated in physics — describes digital image processing as a new tool for scientific research. The book starts with the essentials of image processing and leads — in selected areas — to the state-of-the art. This approach gives an insight as to how image processing really works. The selection of the material is guided by the needs of a researcher who wants to apply image processing techniques in his or her field. In this sense, this book tries to offer an integral view of image processing from image acquisition to the extraction of the data of interest. Many concepts and mathematical tools which find widespread application in natural sciences are also applied in digital image processing. Such analogies are pointed out, since they provide an easy access to many complex problems in digital image processing for readers with a general background in natural sciences. The discussion of the general concepts is supplemented with examples from applications on PC-based image processing systems and ready-to-use implementations of important algorithms.

I am deeply indebted to the many individuals who helped me to write this book. I do this by tracing its history. In the early 1980s, when I worked on the physics of small-scale air-sea interaction at the Institute of Environmental Physics at Heidelberg University, it became obvious that these complex phenomena could not be adequately treated with point measuring probes. Consequently, a number of area extended measuring techniques were developed. Then I searched for techniques to extract the physically relevant data from the images and sought for colleagues with experience in digital image processing. The first contacts were established with the Institute for Applied Physics at Heidelberg University and the German Cancer Research Center in Heidelberg. I would like to thank Prof. Dr. J. Bille, Dr. J. Dengler and Dr. M. Schmidt cordially for many eye-opening conversations and their cooperation.

I would also like to thank Prof. Dr. K. O. Münnich, director of the Institute for Environmental Physics. From the beginning, he was open-minded about new ideas on the application of digital image processing techniques in environmental physics. It is due to his farsightedness and substantial support that the research group “Digital Image Processing in Environmental Physics” could develop so fruitfully at his institute. Many of the examples shown in this book are taken from my research at Heidelberg University and the Scripps Institution of Oceanography. I gratefully acknowledge financial support for this research from the German Science Foundation, the European Community, the US National Science Foundation, and the US Office of Naval Research.

La Jolla, California, and Heidelberg, spring 1991 Bernd Jähne

Contents

I Foundation

1 Applications and Tools 3
- 1.1 A Tool for Science and Technique . . . 3
- 1.2 Examples of Applications . . . 4
- 1.3 Hierarchy of Image Processing Operations . . . 15
- 1.4 Image Processing and Computer Graphics . . . 17
- 1.5 Cross-disciplinary Nature of Image Processing . . . 17
- 1.6 Human and Computer Vision . . . 18
- 1.7 Components of an Image Processing System . . . 21
- 1.8 Further Readings‡ . . . 26

2 Image Representation 29
- 2.1 Introduction . . . 29
- 2.2 Spatial Representation of Digital Images . . . 29
- 2.3 Wave Number Space and Fourier Transform . . . 39
- 2.4 Discrete Unitary Transforms‡ . . . 60
- 2.5 Fast Algorithms for Unitary Transforms . . . 65
- 2.6 Further Readings‡ . . . 76

3 Random Variables and Fields 77
- 3.1 Introduction . . . 77
- 3.2 Random Variables . . . 79
- 3.3 Multiple Random Variables . . . 82
- 3.4 Probability Density Functions . . . 87
- 3.5 Stochastic Processes and Random Fields‡ . . . 93
- 3.6 Further Readings‡ . . . 97

4 Neighborhood Operations 99
- 4.1 Basic Properties and Purpose . . . 99
- 4.2 Linear Shift-Invariant Filters† . . . 102
- 4.3 Recursive Filters‡ . . . 115
- 4.4 Rank Value Filtering . . . 123
- 4.5 Further Readings‡ . . . 124

5 Multiscale Representation 125
5.1 Scale . . . 125
5.2 Scale Space† . . . 128
5.3 Multigrid Representations . . . 136
5.4 Further Readings‡ . . . 142

II Image Formation and Preprocessing

6 Quantitative Visualization 145
6.1 Introduction . . . 145
6.2 Waves and Particles . . . 147
6.3 Radiometry, Photometry, Spectroscopy, and Color . . . 153
6.4 Interactions of Radiation with Matter‡ . . . 162
6.5 Further Readings‡ . . . 175

7 Image Formation 177
7.1 Introduction . . . 177
7.2 World and Camera Coordinates . . . 177
7.3 Ideal Imaging: Perspective Projection† . . . 181
7.4 Real Imaging . . . 185
7.5 Radiometry of Imaging . . . 191
7.6 Linear System Theory of Imaging† . . . 194
7.7 Further Readings‡ . . . 203

8 3-D Imaging 205
8.1 Basics . . . 205
8.2 Depth from Triangulation . . . 208
8.3 Depth from Time-of-Flight . . . 217
8.4 Depth from Phase: Interferometry . . . 217
8.5 Shape from Shading† . . . 218
8.6 Depth from Multiple Projections: Tomography . . . 224
8.7 Further Readings‡ . . . 231

9 Digitization, Sampling, Quantization 233
9.1 Definition and Effects of Digitization . . . 233
9.2 Image Formation, Sampling, Windowing . . . 235
9.3 Reconstruction from Samples† . . . 239
9.4 Quantization . . . 243
9.5 Further Readings‡ . . . 244

10 Pixel Processing 245
10.1 Introduction . . . 245
10.2 Homogeneous Point Operations . . . 246
10.3 Inhomogeneous Point Operations† . . . 256
10.4 Multichannel Point Operations‡ . . . 263
10.5 Geometric Transformations . . . 265
10.6 Interpolation† . . . 269
10.7 Further Readings‡ . . . 280

III Feature Extraction

11 Averaging 283
11.1 Introduction . . . 283
11.2 General Properties of Averaging Filters . . . 283
11.3 Box Filter . . . 286
11.4 Binomial Filter . . . 290
11.5 Filters as Networks‡ . . . 296
11.6 Efficient Large-Scale Averaging‡ . . . 298
11.7 Nonlinear Averaging . . . 307
11.8 Averaging in Multichannel Images‡ . . . 313
11.9 Further Readings‡ . . . 314

12 Edges 315
12.1 Introduction . . . 315
12.2 General Properties of Edge Filters . . . 316
12.3 Gradient-Based Edge Detection† . . . 319
12.4 Edge Detection by Zero Crossings . . . 328
12.5 Regularized Edge Detection . . . 330
12.6 Edges in Multichannel Images‡ . . . 335
12.7 Further Readings‡ . . . 337

13 Simple Neighborhoods 339
13.1 Introduction . . . 339
13.2 Properties of Simple Neighborhoods . . . 340
13.3 First-Order Tensor Representation† . . . 344
13.4 Local Wave Number and Phase‡ . . . 358
13.5 Tensor Representation by Quadrature Filter Sets‡ . . . 368
13.6 Further Readings‡ . . . 374

14 Motion 375
14.1 Introduction . . . 375
14.2 Basics . . . 376
14.3 First-Order Differential Methods† . . . 391
14.4 Tensor Methods . . . 398
14.5 Second-Order Differential Methods‡ . . . 403
14.6 Correlation Methods . . . 407
14.7 Phase Method‡ . . . 409
14.8 Further Readings‡ . . . 412

15 Texture 413
15.1 Introduction . . . 413
15.2 First-Order Statistics . . . 416
15.3 Rotation and Scale Variant Texture Features . . . 420
15.4 Further Readings‡ . . . 424

IV Image Analysis

16 Segmentation 427
16.1 Introduction . . . 427
16.2 Pixel-Based Segmentation . . . 427
16.3 Edge-Based Segmentation . . . 431
16.4 Region-Based Segmentation . . . 432
16.5 Model-Based Segmentation . . . 436
16.6 Further Readings‡ . . . 439

17 Regularization and Modeling 441
17.1 Unifying Local Analysis and Global Knowledge . . . 441
17.2 Purpose and Limits of Models . . . 442
17.3 Variational Image Modeling† . . . 444
17.4 Controlling Smoothness . . . 451
17.5 Diffusion Models . . . 455
17.6 Discrete Inverse Problems† . . . 460
17.7 Network Models‡ . . . 469
17.8 Inverse Filtering . . . 473
17.9 Further Readings‡ . . . 480

18 Morphology 481
18.1 Introduction . . . 481
18.2 Neighborhood Operations on Binary Images . . . 481
18.3 General Properties . . . 483
18.4 Composite Morphological Operators . . . 486
18.5 Further Readings‡ . . . 494

19 Shape Presentation and Analysis 495
19.1 Introduction . . . 495
19.2 Representation of Shape . . . 495
19.3 Moment-Based Shape Features . . . 500
19.4 Fourier Descriptors . . . 502
19.5 Shape Parameters . . . 508
19.6 Further Readings‡ . . . 511

20 Classification 513
20.1 Introduction . . . 513
20.2 Feature Space . . . 516
20.3 Simple Classification Techniques . . . 523
20.4 Further Readings‡ . . . 528

V Reference Part

A Reference Material 531

B Notation 555

Bibliography 563

Index 575

Part I

Foundation

1 Applications and Tools

1.1 A Tool for Science and Technique

From the beginning of science, visual observation has played a major role. At that time, the only way to document the results of an experiment was by verbal description and manual drawings. The next major step was the invention of *photography* which enabled results to be documented objectively. Three prominent examples of scientific applications of photography are *astronomy*, *photogrammetry*, and *particle physics*. Astronomers were able to measure positions and magnitudes of stars and photogrammeters produced topographic maps from aerial images. Searching through countless images from hydrogen bubble chambers led to the discovery of many elementary particles in physics. These manual evaluation procedures, however, were time consuming. Some semi- or even fully automated optomechanical devices were designed. However, they were adapted to a single specific purpose. This is why quantitative evaluation of images did not find widespread application at that time. Generally, images were only used for documentation, qualitative description, and illustration of the phenomena observed.

Nowadays, we are in the middle of a second revolution sparked by the rapid progress in video and computer technology. Personal computers and workstations have become powerful enough to process image data. As a result, multimedia software and hardware is becoming standard for the handling of images, image sequences, and even 3-D visualization. The technology is now available to any scientist or engineer. In consequence, image processing has expanded and is further rapidly expanding from a few specialized applications into a standard scientific tool. Image processing techniques are now applied to virtually all the natural sciences and technical disciplines.

A simple example clearly demonstrates the power of visual information. Imagine you had the task of writing an article about a new technical system, for example, a new type of solar power plant. It would take an enormous effort to describe the system if you could not include images and technical drawings. The reader of your imageless article would also have a frustrating experience. He or she would spend a lot of time trying to figure out how the new solar power plant worked and might end up with only a poor picture of what it looked like.

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2
Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

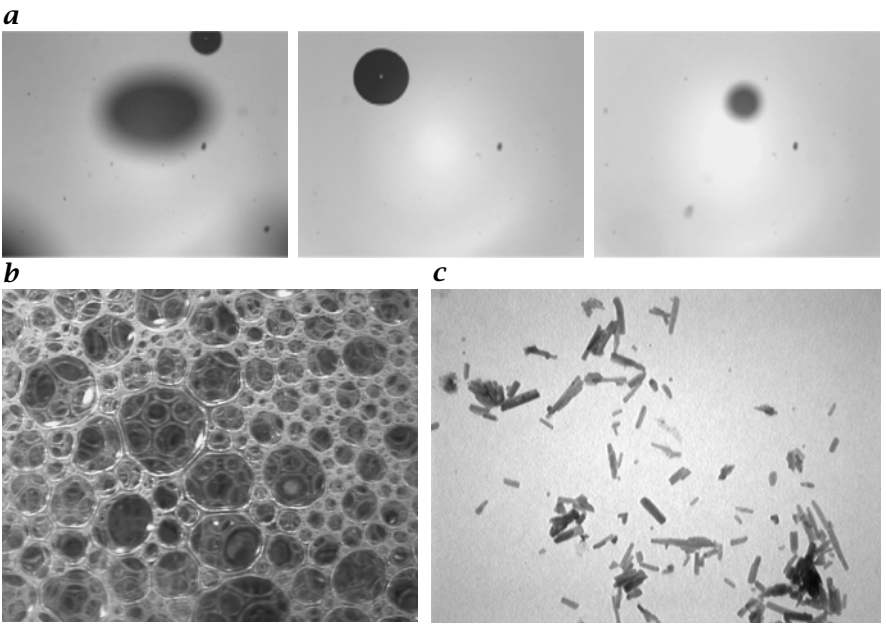

Figure 1.1: *Measurement of particles with imaging techniques:* ***a*** *Bubbles submerged by breaking waves using a telecentric illumination and imaging system; from Geißler and Jähne [50].* ***b*** *Soap bubbles.* ***c*** *Electron microscopy of color pigment particles (courtesy of Dr. Klee, Hoechst AG, Frankfurt).*

Technical drawings and photographs of the solar power plant would be of enormous help for readers of your article. They would immediately have an idea of the plant and could study details in the drawings and photographs which were not described in the text, but which caught their attention. Pictorial information provides much more detail, a fact which can be precisely summarized by the saying that “a picture is worth a thousand words”. Another observation is of interest. If the reader later heard of the new solar plant, he or she could easily recall what it looked like, the object “solar plant” being instantly associated with an image.

1.2 Examples of Applications

In this section, examples for scientific and technical applications of digital image processing are discussed. The examples demonstrate that image processing enables complex phenomena to be investigated, which could not be adequately accessed with conventional measuring techniques.

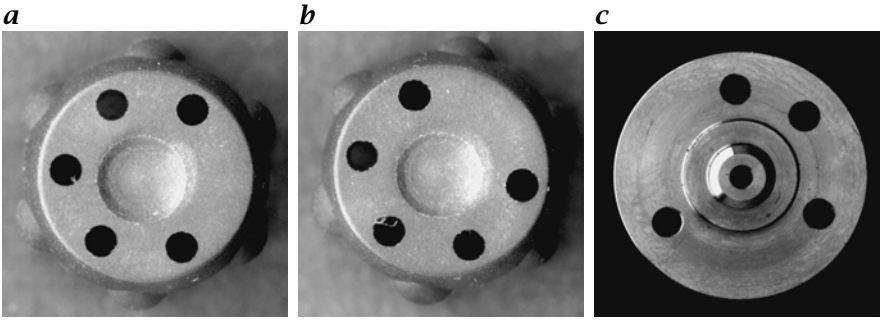

Figure 1.2: *Industrial parts that are checked by a visual inspection system for the correct position and diameter of holes (courtesy of Martin von Brocke, Robert Bosch GmbH).*

1.2.1 Counting and Gauging

A classic task for digital image processing is counting particles and measuring their size distribution. Figure 1.1 shows three examples with very different particles: gas bubbles submerged by breaking waves, soap bubbles, and pigment particles. The first challenge with tasks like this is to find an imaging and illumination setup that is well adapted to the measuring problem. The bubble images in Fig. 1.1a are visualized by a *telecentric* illumination and imaging system. With this setup, the principle rays are parallel to the optical axis. Therefore the size of the imaged bubbles does not depend on their distance. The sampling volume for concentration measurements is determined by estimating the degree of blurring in the bubbles.

It is much more difficult to measure the shape of the soap bubbles shown in Fig. 1.1b, because they are transparent. Therefore, deeper lying bubbles superimpose the image of the bubbles in the front layer. Moreover, the bubbles show deviations from a circular shape so that suitable parameters must be found to describe their shape.

A third application is the measurement of the size distribution of color pigment particles. This significantly influences the quality and properties of paint. Thus, the measurement of the distribution is an important quality control task. The image in Fig. 1.1c taken with a transmission electron microscope shows the challenge of this image processing task. The particles tend to cluster. Consequently, these clusters have to be identified, and — if possible — to be separated in order not to bias the determination of the size distribution.

Almost any product we use nowadays has been checked for defects by an automatic *visual inspection* system. One class of tasks includes the checking of correct sizes and positions. Some example images are

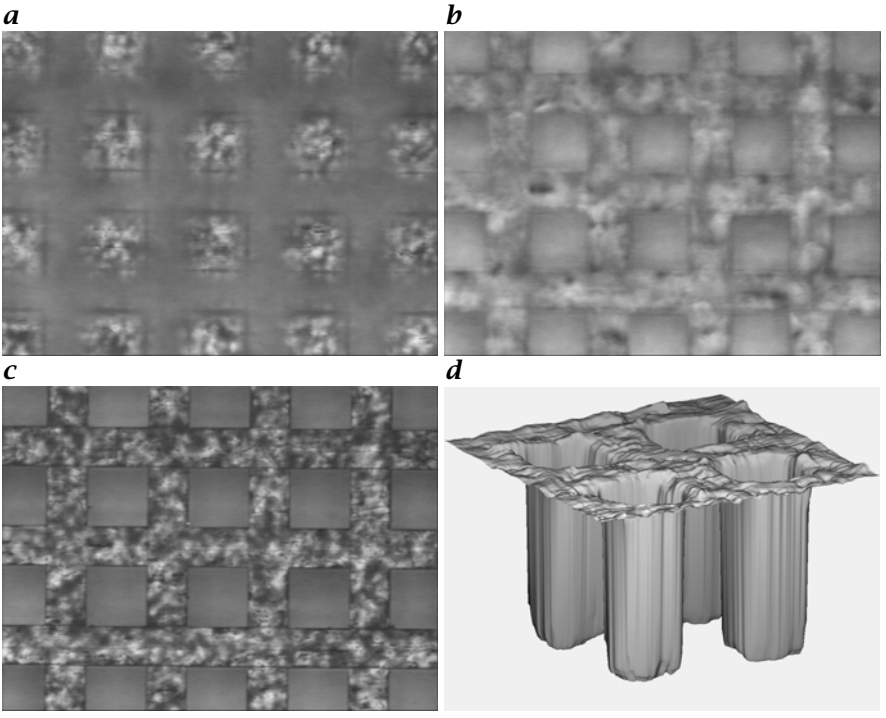

Figure 1.3: *Focus series of a press form of PMMA with narrow rectangular holes imaged with a confocal technique using statistically distributed intensity patterns. The images are focused on the following depths measured from the bottom of the holes:* ***a*** *16 µm,* ***b*** *480 µm, and* ***c*** *620 µm (surface of form).* ***d*** *3-D reconstruction. From Scheuermann et al. [163].*

shown in Fig. 1.2. Here the position, diameter, and roundness of the holes is checked. Figure 1.2c illustrates that it is not easy to illuminate metallic parts. The edge of the hole on the left is partly bright and thus it is more difficult to detect and to measure the holes correctly.

1.2.2 Exploring 3-D Space

In images, 3-D scenes are projected on a 2-D image plane. Thus the depth information is lost and special imaging techniques are required to retrieve the topography of surfaces or volumetric images. In recent years, a large variety of range imaging and volumetric imaging techniques have been developed. Therefore image processing techniques are also applied to *depth maps* and *volumetric images*.

Figure 1.3 shows the reconstruction of a press form for microstructures that has been imaged by a special type of confocal microscopy [163]. The form is made out of PMMA, a semi-transparent plastic ma-

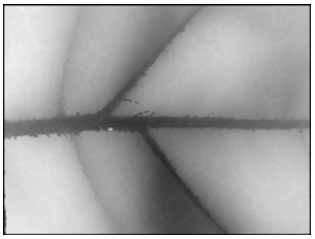

Figure 1.4: *Depth map of a plant leaf measured by optical coherency tomography (courtesy of Jochen Restle, Robert Bosch GmbH).*

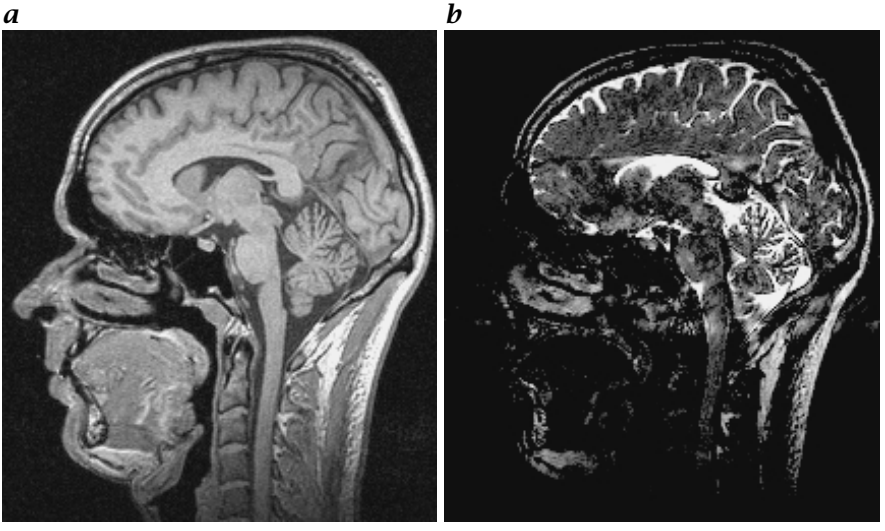

Figure 1.5: *Magnetic resonance image of a human head:* ***a*** *T1 image;* ***b*** *T2 image (courtesy of Michael Bock, DKFZ Heidelberg).*

terial with a smooth surface, so that it is almost invisible in standard microscopy. The form has narrow, 500 μm deep rectangular holes.

In order to make the transparent material visible, a statistically distributed pattern is projected through the microscope optics onto the focal plane. This pattern only appears sharp on parts that lie in the focal plane. The pattern gets more blurred with increasing distance from the focal plane. In the focus series shown in Fig. 1.3, it can be seen that first the patterns of the material in the bottom of the holes become sharp (Fig. 1.3a), then after moving the object away from the optics, the final image focuses at the surface of the form (Fig. 1.3c). The depth of the surface can be reconstructed by searching for the position of maximum contrast for each pixel in the focus series (Fig. 1.3d).

Figure 1.4 shows the depth map of a plant leaf that has been imaged with another modern optical 3-D measuring technique known as *white-*

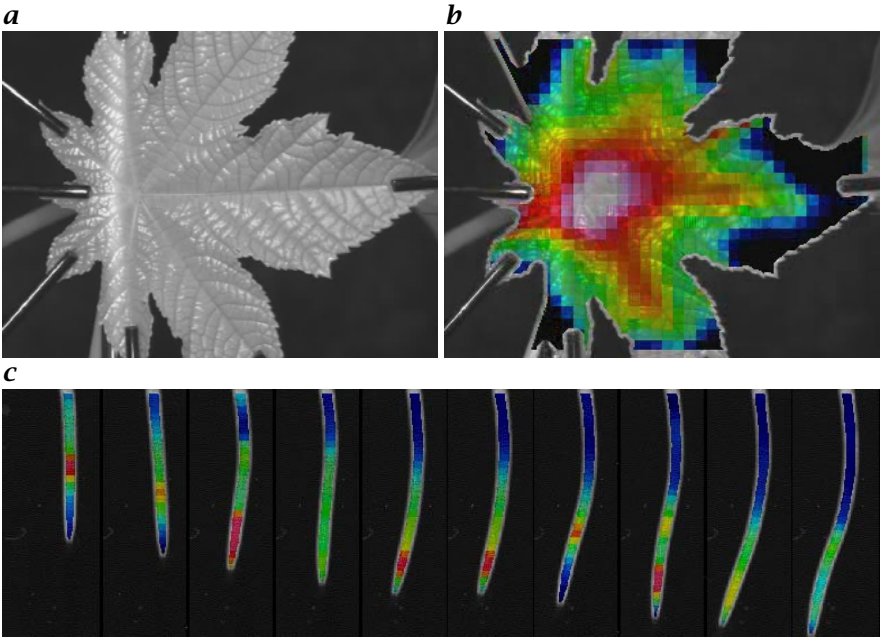

Figure 1.6: *Growth studies in botany:* ***a*** *Rizinus plant leaf;* ***b*** *map of growth rate;* ***c*** *Growth of corn roots (courtesy of Uli Schurr and Stefan Terjung, Institute of Botany, University of Heidelberg).*

light interferometry or *coherency radar*. It is an interferometric technique that uses light with a coherency length of only a few wavelengths. Thus interference patterns occur only with very short path differences in the interferometer. This effect can be utilized to measure distances with an accuracy in the order of a wavelength of light used.

Magnetic resonance imaging (*MR*) is an example of a modern volumetric imaging technique, which we can use to look into the interior of 3-D objects. In contrast to x-ray tomography, it can distinguish different tissues such as gray and white brain tissues. Magnetic resonance imaging is a very flexible technique. Depending on the parameters used, quite different material properties can be visualized (Fig. 1.5).

1.2.3 Exploring Dynamic Processes

The exploration of dynamic processes is possible by analyzing *image sequences*. The enormous potential of this technique is illustrated with a number of examples in this section.

In botany, a central topic is the study of the growth of plants and the mechanisms controlling growth processes. Figure 1.6a shows a Rizinus plant leaf from which a map of the growth rate (percent increase of

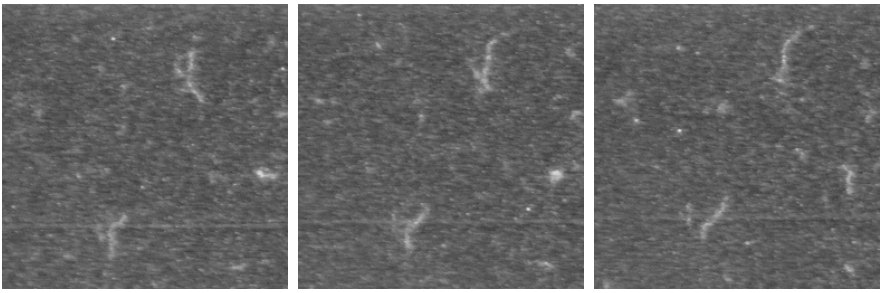

Figure 1.7: *Motility assay for motion analysis of motor proteins (courtesy of Dietmar Uttenweiler, Institute of Physiology, University of Heidelberg).*

area per unit time) has been determined by a time-lapse image sequence where about every minute an image was taken. This new technique for growth rate measurements is sensitive enough for area-resolved measurements of the diurnal cycle.

Figure 1.6c shows an image sequence (from left to right) of a growing corn root. The gray scale in the image indicates the growth rate, which is largest close to the tip of the root.

In science, images are often taken at the limit of the technically possible. Thus they are often plagued by high noise levels. Figure 1.7 shows fluorescence-labeled motor proteins that a moving on a plate covered with myosin molecules in a so-called *motility assay*. Such an assay is used to study the molecular mechanisms of muscle cells. Despite the high noise level, the motion of the filaments is apparent. However, automatic motion determination with such noisy image sequences is a demanding task that requires sophisticated image sequence analysis techniques.

The next example is taken from oceanography. The small-scale processes that take place in the vicinity of the ocean surface are very difficult to measure because of undulation of the surface by waves. Moreover, point measurements make it impossible to infer the 2-D structure of the waves at the water surface. Figure 1.8 shows a space-time image of short wind waves. The vertical coordinate is a spatial coordinate in the wind direction and the horizontal coordinate the time. By a special illumination technique based on the *shape from shading* paradigm (Section 8.5.3), the along-wind slope of the waves has been made visible. In such a spatiotemporal image, motion is directly visible by the inclination of lines of constant gray scale. A horizontal line marks a static object. The larger the angle to horizontal axis, the faster the object is moving. The image sequence gives a direct insight into the complex nonlinear dynamics of wind waves. A fast moving large wave modulates the motion of shorter waves. Sometimes the short waves move with the same speed (bound waves), but mostly they are significantly slower showing large modulations in the phase speed and amplitude.

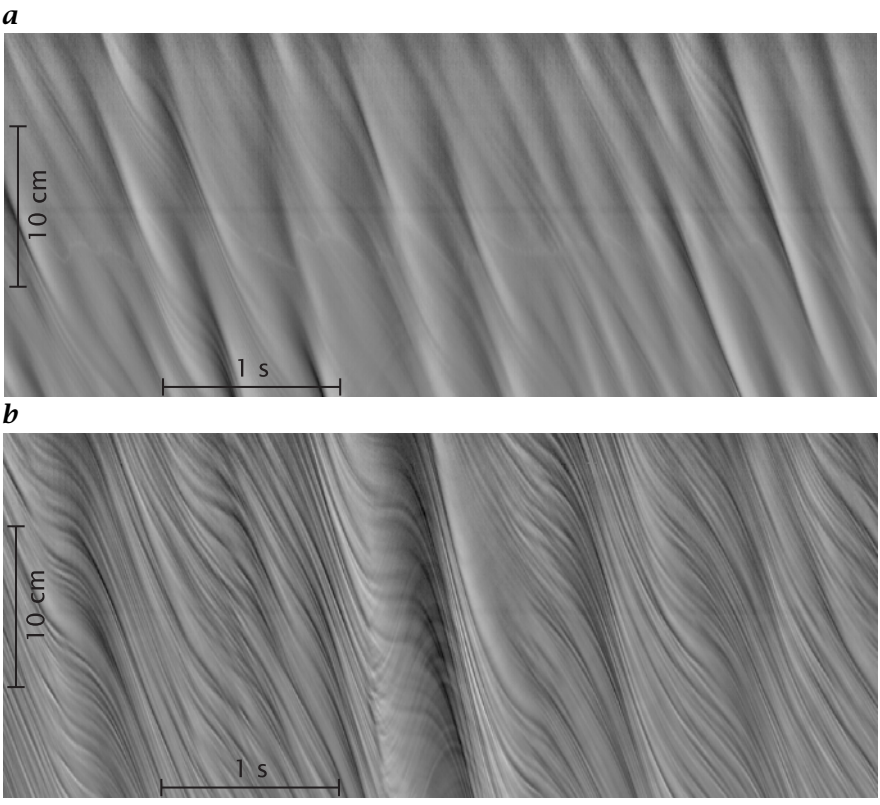

Figure 1.8: *A space-time image of short wind waves at a wind speed of* ***a*** *2.5 and* ***b*** *7.5 m/s. The vertical coordinate is the spatial coordinate in wind direction, the horizontal coordinate the time.*

The last example of image sequences is on a much larger spatial and temporal scale. Figure 1.9 shows the annual cycle of the tropospheric column density of NO_2. NO_2 is one of the most important trace gases for the atmospheric ozone chemistry. The main sources for tropospheric NO_2 are industry and traffic, forest and bush fires (biomass burning), microbiological soil emissions, and lighting. Satellite imaging allows for the first time the study of the regional distribution of NO_2 and the identification of the sources and their annual cycles.

The data have been computed from spectroscopic images obtained from the GOME instrument of the ERS2 satellite. At each pixel of the images a complete spectrum with 4000 channels in the ultraviolet and visible range has been taken. The total atmospheric column density of the NO_2 concentration can be determined by the characteristic absorption spectrum that is, however, superimposed by the absorption spectra of other trace gases. Therefore, a complex nonlinear regression analy-

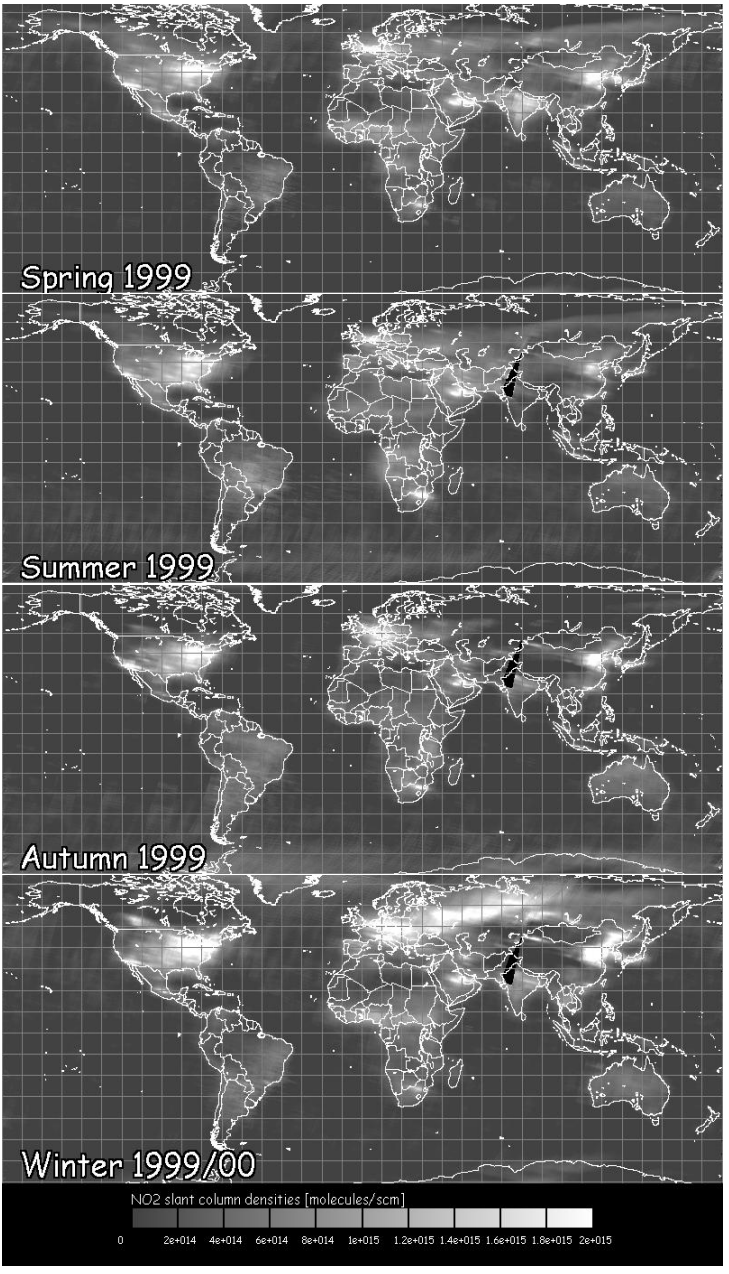

Figure 1.9: *Maps of tropospheric NO_2 column densities showing four three-month averages from 1999 (courtesy of Mark Wenig, Institute for Environmental Physics, University of Heidelberg).*

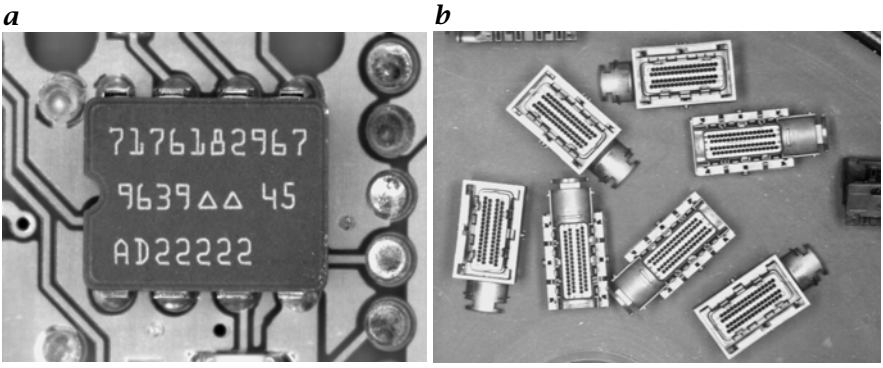

Figure 1.10: *Industrial inspection tasks:* ***a*** *Optical character recognition.* ***b*** *Connectors (courtesy of Martin von Brocke, Robert Bosch GmbH).*

sis is required. Furthermore, the stratospheric column density must be subtracted by suitable image processing algorithms.

The resulting maps of tropospheric NO_2 column densities in Fig. 1.9 clearly show a lot of interesting detail. Most emissions are clearly related to industrialized countries. They show a clear annual cycle in the Northern hemisphere with a maximum in the winter.

1.2.4 Classification

Another important task is the classification of objects observed in images. The classical example of classification is the recognition of characters (*optical character recognition* or short *OCR*). Figure 1.10a shows a typical industrial OCR application, the recognition of a label on an integrated circuit. Object classification includes also the recognition of different possible positioning of objects for correct handling by a robot. In Fig. 1.10b, connectors are placed in random orientation on a conveyor belt. For proper pick up and handling, whether the front or rear side of the connector is seen must also be detected.

The classification of defects is another important application. Figure 1.11 shows a number of typical errors in the inspection of integrated circuits: an incorrectly centered surface mounted resistor (Fig. 1.11a), and broken or missing bond connections (Fig. 1.11b–f).

The application of classification is not restricted to industrial tasks. Figure 1.12 shows some of the most distant galaxies ever imaged by the Hubble telescope. The galaxies have to be separated into different classes due to their shape and color and have to be distinguished from other objects, e. g., stars.

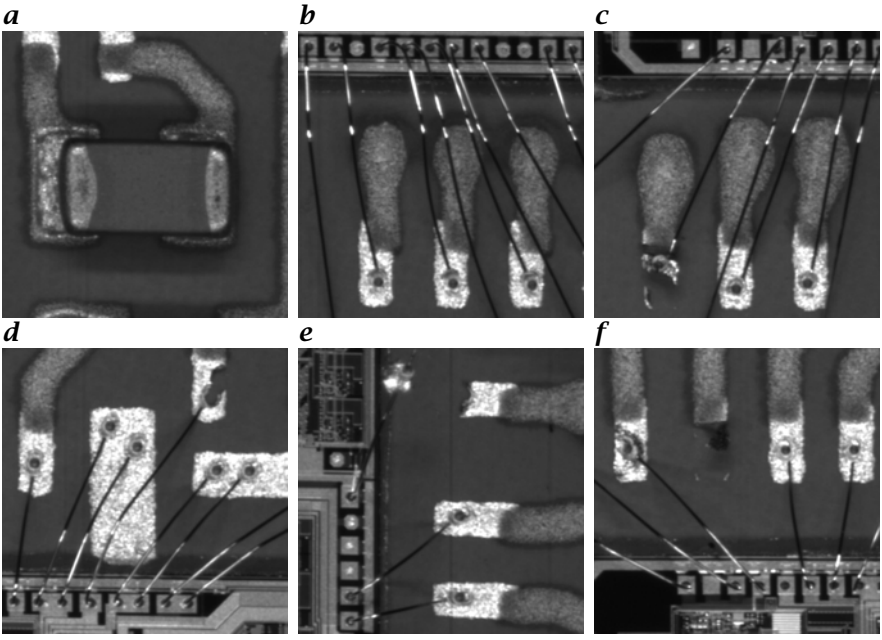

Figure 1.11: *Errors in soldering and bonding of integrated circuits. Courtesy of Florian Raisch, Robert Bosch GmbH).*

Figure 1.12: *Hubble deep space image: classification of distant galaxies (http://hubblesite.org/).*

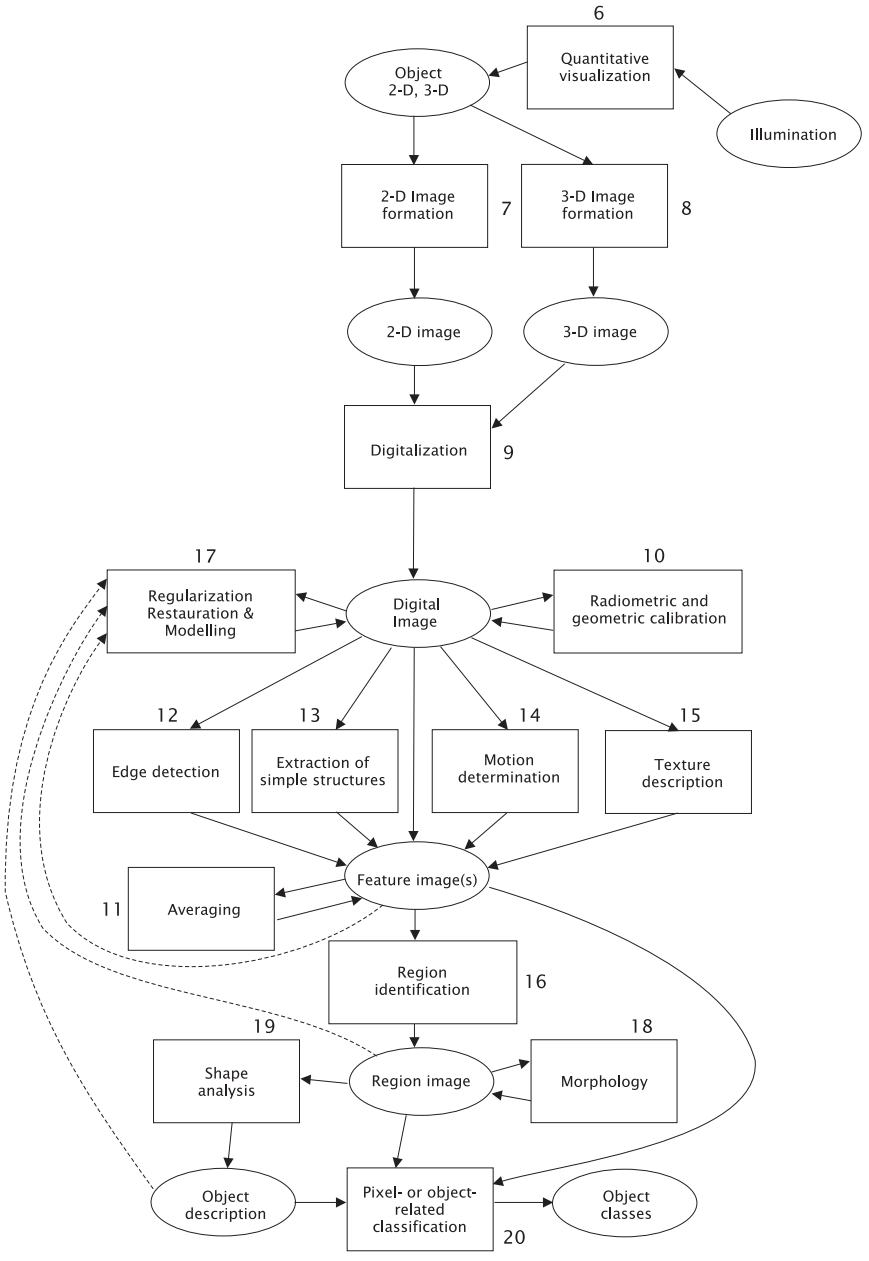

Figure 1.13: *A hierarchy of digital image processing tasks from image formation to image comprehension. The numbers by the boxes indicate the corresponding chapters of this book.*

1.3 Hierarchy of Image Processing Operations

Image processing is not a one-step process. We are able to distinguish between several steps which must be performed one after the other until we can extract the data of interest from the observed scene. In this way a *hierarchical processing* scheme is built up as sketched in Fig. 1.13. The figure gives an overview of the different phases of image processing, together with a summary outline of this book.

Image processing begins with the capture of an image with a suitable, not necessarily optical, acquisition system. In a technical or scientific application, we may choose to select an appropriate imaging system. Furthermore, we can set up the illumination system, choose the best wavelength range, and select other options to capture the object feature of interest in the best way in an image (Chapter 6). 2-D and 3-D image formation are discussed in Chapters 7 and 8, respectively. Once the image is sensed, it must be brought into a form that can be treated with digital computers. This process is called *digitization* and is discussed in Chapter 9.

The first steps of digital processing may include a number of different operations and are known as *image preprocessing*. If the sensor has nonlinear characteristics, these need to be corrected. Likewise, brightness and contrast of the image may require improvement. Commonly, too, coordinate transformations are needed to restore geometrical distortions introduced during image formation. Radiometric and geometric corrections are elementary pixel processing operations that are discussed in Chapter 10.

A whole chain of processing steps is necessary to analyze and identify objects. First, adequate filtering procedures must be applied in order to distinguish the objects of interest from other objects and the background. Essentially, from an image (or several images), one or more *feature images* are extracted. The basic tools for this task are averaging (Chapter 11), edge detection (Chapter 12), the analysis of simple neighborhoods (Chapter 13) and complex patterns known in image processing as *texture* (Chapter 15). An important feature of an object is also its *motion*. Techniques to detect and determine motion are discussed in Chapter 14.

Then the object has to be separated from the background. This means that regions of constant features and discontinuities must be identified by *segmentation* (Chapter 16). This can be an easy task if an object is well distinguished from the background by some local features. This is, however, not often the case. Then more sophisticated segmentation techniques are required (Chapter 17). These techniques use various optimization strategies to minimize the deviation between the image data and a given model function incorporating the knowledge about the objects in the image.

The same mathematical approach can be used for other image processing tasks. Known disturbances in the image, for instance caused by a defocused optics, motion blur, errors in the sensor, or errors in the transmission of image signals, can be corrected (*image restoration*). Images can be reconstructed from indirect imaging techniques such as *tomography* that deliver no direct image (*image reconstruction*).

Now that we know the geometrical shape of the object, we can use morphological operators to analyze and modify the shape of objects (Chapter 18) or extract further information such as the mean gray value, the area, perimeter, and other parameters for the form of the object (Chapter 19). These parameters can be used to classify objects (*classification*, Chapter 20). Character recognition in printed and handwritten text is an example of this task.

While it appears logical to part a complex task such as image processing into a succession of simple subtasks, it is not obvious that this strategy works at all. Why? Let us discuss a simple example. We want to find an object that differs in its gray value only slightly from the background in a noisy image. In this case, we cannot simply take the gray value to differentiate the object from the background. Averaging of neighboring image points can reduce the noise level. At the edge of the object, however, background and object points are averaged, resulting in false mean values. If we knew the edge, averaging could be stopped at the edge. But we can determine the edges only after averaging because only then are the gray values of the object sufficiently different from the background. We may hope to escape this circular argument by an iterative approach. We just apply the averaging and make a first estimate of the edges of the object. We then take this first estimate to refine the averaging at the edges, recalculate the edges and so on. It remains to be studied in detail, however, whether this iteration converges at all, and if it does, whether the limit is correct.

In any case, the discussed example suggests that more difficult image processing tasks require feedback. Advanced processing steps give parameters back to preceding processing steps. Then the processing is not linear along a chain but may iteratively loop back several times. Figure 1.13 shows some possible feedbacks. The feedback may include non-image processing steps. If an image processing task cannot be solved with a given image, we may decide to change the illumination, zoom closer to an object of interest or to observe it under a more suitable view angle. This type of approach is known as *active vision*. In the framework of an intelligent system exploring its environment by its senses we may also speak of an *action-perception cycle*.

1.4 Image Processing and Computer Graphics

For some time now, image processing and *computer graphics* have been treated as two different areas. Knowledge in both areas has increased considerably and more complex problems can now be treated. Computer graphics is striving to achieve *photorealistic* computer-generated images of three-dimensional scenes, while image processing is trying to reconstruct one from an image actually taken with a camera. In this sense, *image processing* performs the inverse procedure to that of computer graphics. In computer graphics we start with knowledge of the shape and features of an object — at the bottom of Fig. 1.13 — and work upwards until we get a two-dimensional image. To handle image processing or computer graphics, we basically have to work from the same knowledge. We need to know the interaction between illumination and objects, how a three-dimensional scene is projected onto an image plane, etc.

There are still quite a few differences between an image processing and a graphics workstation. But we can envisage that, when the similarities and interrelations between computer graphics and image processing are better understood and the proper hardware is developed, we will see some kind of general-purpose workstation in the future which can handle computer graphics as well as image processing tasks. The advent of multimedia, i. e., the integration of text, images, sound, and movies, will further accelerate the unification of computer graphics and image processing. The term "*visual computing*" has been coined in this context [58].

1.5 Cross-disciplinary Nature of Image Processing

By its very nature, the science of image processing is cross-disciplinary in several aspects. First, image processing incorporates concepts from various sciences. Before we can process an image, we need to know how the digital signal is related to the features of the imaged objects. This includes various physical processes from the interaction of radiation with matter to the geometry and radiometry of imaging. An imaging sensor converts the incident irradiance in one or the other way into an electric signal. Next, this signal is converted into digital numbers and processed by a digital computer to extract the relevant data. In this chain of processes (see also Fig. 1.13) many areas from *physics*, *computer science* and *mathematics* are involved including among others, optics, solid state physics, chip design, computer architecture, algebra, analysis, statistics, algorithm theory, graph theory, system theory, and numerical mathematics. From an engineering point of view, contributions from *optical engineering*, *electrical engineering*, *photonics*, and *software engineering* are required.

Image processing has a partial overlap with other disciplines. Image processing tasks can partly be regarded as a measuring problem, which is part of the science of *metrology*. Likewise, *pattern recognition* tasks are incorporated in image processing in a similar way as in *speech processing*. Other disciplines with similar connections to image processing are the areas of *neural networks*, *artificial intelligence*, and *visual perception*. Common to these areas is their strong link to biological sciences.

When we speak of *computer vision*, we mean a computer system that performs the same task as a biological vision system to "discover from images what is present in the world, and where it is" [120]. In contrast, the term *machine vision* is used for a system that performs a vision task such as checking the sizes and completeness of parts in a manufacturing environment. For many years, a vision system has been regarded just as a passive observer. As with biological vision systems, a computer vision system can also actively explore its surroundings by, e. g., moving around and adjusting its angle of view. This, we call *active vision*.

There are numerous special disciplines that for historical reasons developed partly independently of the main stream in the past. One of the most prominent disciplines is *photogrammetry* (measurements from photographs; main applications: mapmaking and surveying). Other areas are *remote sensing* using aerial and satellite images, *astronomy*, and *medical imaging*.

The second important aspect of the cross-disciplinary nature of image processing is its widespread application. There is almost no field in natural sciences or technical disciplines where image processing is not applied. As we have seen from the examples in Section 1.2, it has gained crucial importance in several application areas. The strong links to so many application areas provide a fertile ground for further rapid progress in image processing because of the constant inflow of techniques and ideas from an ever-increasing host of application areas.

A final cautionary note: a cross-disciplinary approach is not just a nice extension. It is a necessity. Lack of knowledge in either the application area or image processing tools inevitably leads at least to suboptimal solutions and sometimes even to a complete failure.

1.6 Human and Computer Vision

We cannot think of image processing without considering the *human visual system*. This seems to be a trivial statement, but it has far-reaching consequences. We observe and evaluate the images that we process with our visual system. Without taking this elementary fact into consideration, we may be much misled in the interpretation of images.
The first simple questions we should ask are:

- What intensity differences can we distinguish?

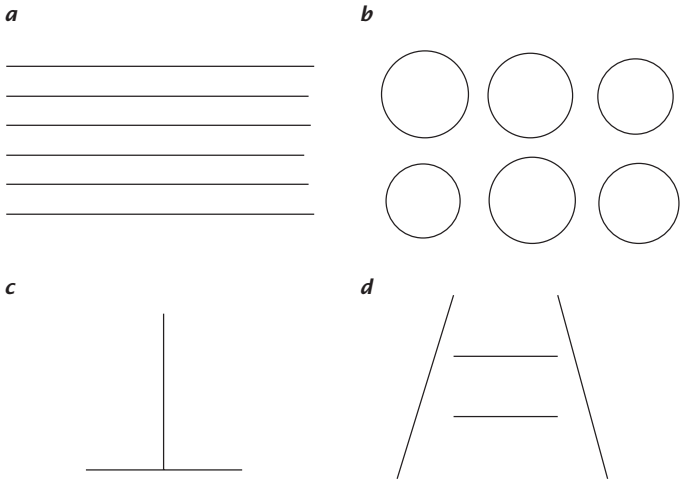

Figure 1.14: *Test images for distance and area estimation:* ***a*** *parallel lines with up to 5 % difference in length;* ***b*** *circles with up to 10 % difference in radius;* ***c*** *the vertical line appears longer, though it has the same length as the horizontal line;* ***d*** *deception by perspective: the upper line (in the background) appears longer than the lower line (in the foreground), though both are equally long.*

- What is the spatial resolution of our eye?
- How accurately can we estimate and compare distances and areas?
- How do we sense colors?
- By which features can we detect and distinguish objects?

It is obvious that a deeper knowledge would be of immense help for computer vision. Here is not the place to give an overview of the human visual system. The intention is rather to make us aware of the elementary relations between human and computer vision. We will discuss diverse properties of the human visual system in the appropriate chapters. Here, we will make only some introductory remarks. A detailed comparison of human and computer vision can be found in Levine [109]. An excellent up-to-date reference to human vision is also the monograph by Wandell [193].

The reader can perform some experiments by himself. Figure 1.14 shows several test images concerning the question of estimation of distance and area. He will have no problem in seeing even small changes in the length of the parallel lines in Fig. 1.14a. A similar area comparison with circles is considerably more difficult (Fig. 1.14b). The other examples show how the estimate is biased by the context of the image. Such phenomena are known as *optical illusions*. Two examples of estimates for length are shown in Fig. 1.14c, d. These examples show

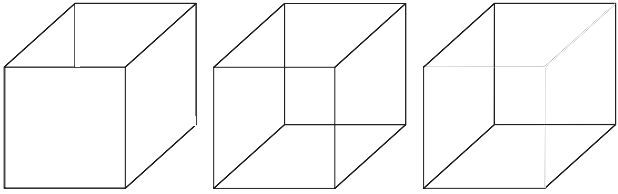

Figure 1.15: *Recognition of three-dimensional objects: three different representations of a cube with identical edges in the image plane.*

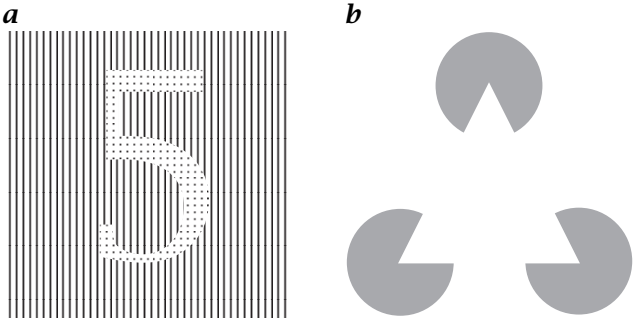

Figure 1.16: ***a*** *Recognition of boundaries between textures;* ***b*** *"interpolation" of object boundaries.*

that the human visual system interprets the context in its estimate of length. Consequently, we should be very careful in our visual estimates of lengths and areas in images.

The second topic is that of the recognition of objects in images. Although Fig. 1.15 contains only a few lines and is a planar image not containing any direct information on depth, we immediately recognize a cube in the right and left image and its orientation in space. The only clues from which we can draw this conclusion are the hidden lines and our knowledge about the shape of a cube. The image in the middle, which also shows the hidden lines, is ambivalent. With some training, we can switch between the two possible orientations in space.

Figure 1.16 shows a remarkable feature of the human visual system. With ease we see sharp boundaries between the different textures in Fig. 1.16a and immediately recognize the figure 5. In Fig. 1.16b we identify a white equilateral triangle, although parts of the bounding lines do not exist.

From these few observations, we can conclude that the human visual system is extremely powerful in recognizing objects, but is less well suited for accurate measurements of gray values, distances, and areas.

In comparison, the power of computer vision systems is marginal and should make us feel humble. A digital image processing system can

only perform elementary or well-defined fixed image processing tasks such as real-time quality control in industrial production. A computer vision system has also succeeded in steering a car at high speed on a highway, even with changing lanes. However, we are still worlds away from a universal digital image processing system which is capable of "understanding" images as human beings do and of reacting intelligently and flexibly in real time.

Another connection between human and computer vision is worth noting. Important developments in computer vision have been made through progress in understanding the human visual system. We will encounter several examples in this book: the *pyramid* as an efficient data structure for image processing (Chapter 5), the concept of local orientation (Chapter 13), and motion determination by filter techniques (Chapter 14).

1.7 Components of an Image Processing System

This section briefly outlines the capabilities of modern image processing systems. A general purpose image acquisition and processing system typically consists of four essential components:

1. An image acquisition system. In the simplest case, this could be a CCD camera, a flatbed scanner, or a video recorder.
2. A device known as a frame grabber to convert the electrical signal (normally an analog video signal) of the image acquisition system into a digital image that can be stored.
3. A personal computer or a workstation that provides the processing power.
4. Image processing software that provides the tools to manipulate and analyze the images.

1.7.1 Image Sensors

Digital processing requires images to be obtained in the form of electrical signals. These signals can be digitized into sequences of numbers which then can be processed by a computer. There are many ways to convert images into digital numbers. Here, we will focus on video technology, as it is the most common and affordable approach.

The milestone in image sensing technology was the invention of semiconductor photodetector arrays. There are many types of such sensors, the most common being the *charge coupled device* or *CCD*. Such a sensor consists of a large number of photosensitive elements. During the accumulation phase, each element collects electrical charges, which are generated by absorbed photons. Thus the collected charge is proportional

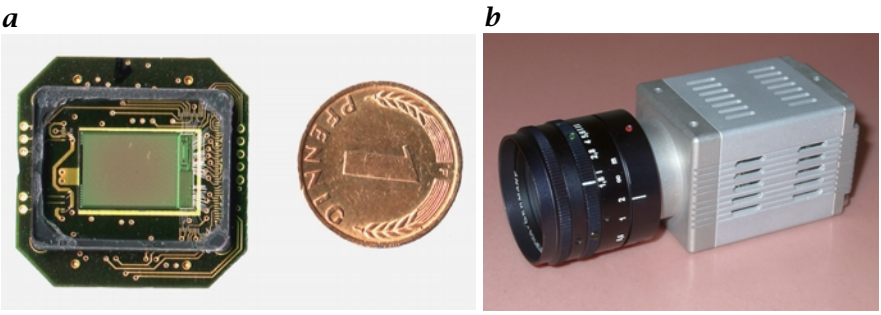

Figure 1.17: *Modern semiconductor cameras:* ***a*** *Complete CMOS camera on a chip with digital and analog output (image courtesy, K. Meier, Kirchhoff-Institute for Physics, University of Heidelberg), [114].* ***b*** *High-end digital 12-bit CCD camera, Pixelfly (image courtesy of PCO GmbH, Germany).*

to the illumination. In the read-out phase, these charges are sequentially transported across the chip from sensor to sensor and finally converted to an electric voltage.

For quite some time, *CMOS image sensors* have been available. But only recently have these devices attracted significant attention because the image quality, especially the uniformity of the sensitivities of the individual sensor elements, now approaches the quality of CCD image sensors. CMOS imagers still do not reach up to the standards of CCD imagers in some features, especially at low illumination levels (higher dark current). They have, however, a number of significant advantages over CCD imagers. They consume significantly less power, subareas can be accessed quickly, and can be added to circuits for image preprocessing and signal conversion. Indeed, it is possible to put a whole camera on a single chip (Fig. 1.17a). Last but not least, CMOS sensors can be manufactured more cheaply and thus open new application areas.

Generally, semiconductor imaging sensors are versatile and powerful devices:

- *Precise and stable geometry.* The individual sensor elements are precisely located on a regular grid. Geometric distortion is virtually absent. Moreover, the sensor is thermally stable in size due to the low linear thermal expansion coefficient of silicon ($2 \cdot 10^{-6}$/K). These features allow precise size and position measurements.
- *Small and rugged.* The sensors are small and insensitive to external influences such as magnetic fields and vibrations.
- *High sensitivity.* The *quantum efficiency*, i. e., the fraction of elementary charges generated per photon, can be close to one (≻ R1 and ≻ R2). However, commercial CCDs at room temperature cannot be used at low light levels because of the thermally generated electrons.

But if CCD devices are cooled down to low temperatures, they can be exposed for hours. Such devices are commonly used in astronomy and are about one hundred times more sensitive than photographic material.

- *Wide variety.* Imaging sensors are available in a wide variety of resolutions and frame rates (≻ R1 and ≻ R2). The largest built CCD sensor as of 2001 originates from Philips. In a modular design with 1k × 1k sensor blocks, they built a 7k × 9k sensor with 12 × 12 μm pixels [60]. Among the fastest high-resolution imagers available is the 1280 × 1024 active-pixel CMOS sensor from Photobit with a peak frame rate of 500 Hz (660 MB/s data rate) [137].
- *Imaging beyond the visible.* Semiconductor imagers are not limited to the visible range of the electromagnetic spectrum. Standard silicon imagers can be made sensitive far beyond the visible wavelength range (400–700 nm) from 200 nm in the *ultraviolet* to 1100 nm in the near *infrared*. In the infrared range beyond 1100 nm, other semiconductors such an GaAs, InSb, HgCdTe are used (≻ R3) since silicon becomes transparent. Towards shorter wavelengths, specially designed silicon imagers can be made sensitive well into the *x-ray* wavelength region.

1.7.2 Image Acquisition and Display

A frame grabber converts the electrical signal from the camera into a digital image that can be processed by a computer. Image display and processing nowadays no longer require any special hardware. With the advent of graphical user interfaces, image display has become an integral part of a personal computer or workstation. Besides the display of gray-scale images with up to 256 shades (8 bit), also true-color images with up to 16.7 million colors (3 channels with 8 bits each), can be displayed on inexpensive PC graphic display systems with a resolution of up to 1600 × 1200 pixels.

Consequently, a modern frame grabber no longer requires its own image display unit. It only needs circuits to digitize the electrical signal from the imaging sensor and to store the image in the memory of the computer. The direct transfer of image data from a frame grabber to the memory (RAM) of a microcomputer has become possible since 1995 with the introduction of fast peripheral bus systems such as the PCI bus. This 32-bit wide and 33 Mhz fast bus has a peak transfer rate of 132 MB/s. Depending on the PCI bus controller on the frame grabber and the chipset on the motherboard of the computer, sustained transfer rates between 15 and 80 MB/s have been reported. This is sufficient to transfer image sequences in real time to the main memory, even for color images and fast frame rate images. The second generation 64-

bit, 66 MHz PCI bus quadruple the data transfer rates to a peak transfer rate of 512 MB/s. Digital cameras that transfer image data directly to the PC via standardized digital interfaces such as *Firewire* (*IEEE 1394*), *Camera link*, or even fast *Ethernet* will further simplify the image input to computers.

The transfer rates to standard hard disks, however, are considerably lower. Sustained transfer rates are typically lower than 10 MB/s. This is inadequate for uncompressed real-time image sequence storage to disk. Real-time transfer of image data with sustained data rates between 10 and 30 MB/s is, however, possible with *RAID arrays*.

1.7.3 Computer Hardware for Fast Image Processing

The tremendous progress of computer technology in the past 20 years has brought digital image processing to the desk of every scientist and engineer. For a general-purpose computer to be useful for image processing, four key demands must be met: high-resolution image display, sufficient memory transfer bandwidth, sufficient storage space, and sufficient computing power. In all four areas, a critical level of performance has been reached that makes it possible to process images on standard hardware. In the near future, it can be expected that general-purpose computers can handle volumetric images and/or image sequences without difficulties. In the following, we will briefly outline these key areas.

General-purpose computers now include sufficient random access memory (RAM) to store multiple images. A 32-bit computer can address up to 4 GB of memory. This is sufficient to handle complex image processing tasks even with large images. Emerging 64-bit computer systems provide enough RAM even for demanding applications with image sequences and volumetric images.

While in the early days of personal computers hard disks had a capacity of just 5–10 MB, nowadays disk systems with more than thousand times more storage capacity (10–60 GB) are standard. Thus, a large number of images can be stored on a disk, which is an important requirement for scientific image processing. For permanent data storage and PC exchange, the *CD-ROM* is playing an important role as a cheap and versatile storage media. One CD can hold up to 600 MB of image data that can be read independent of the operating system on MS Windows, Macintosh, and UNIX platforms. Cheap CD-ROM writers allow anyone to produce CDs. Once cheap *DVD+RW* writers are on the market, a storage media with a even higher capacity of 4.7 GB, compatible to standard *DVD* (*digital video disks*) ROM and video disks, will be available.

Within the short history of microprocessors and personal computers, computing power has increased tremendously. From 1978 to 2001 the clock rate has increased from 4.7 MHz to 1.6 GHz by a factor of 300. The speed of elementary operations such as floating-point addition and mul-

tiplication has increased even more because on modern CPUs these operations have now a throughput of only a few clocks instead of about 100 on early processors. Thus, in less than 25 years, the speed of floating-point computations on a single microprocessor increased more than a factor of 10 000.

Image processing could benefit from this development only partly. On modern 32-bit processors it became increasingly inefficient to transfer and process 8-bit and 16-bit image data. This changed only in 1997 with the integration of multimedia techniques into PCs and workstations. The basic idea of fast image data processing is very simple. It makes use of the 64-bit data paths in modern processors for quick transfer and processing of multiple image data in parallel. This approach to parallel computing is a form of the *single instruction multiple data* (*SIMD*) concept. In 64-bit machines, eight 8-bit, four 16-bit or two 32-bit data can be processed together.

Sun was the first to integrate the SIMD concept into a general-purpose computer architecture with the *visual instruction set* (*VIS*) on the UltraSparc architecture [126]. In January 1997 Intel introduced the *Multimedia Instruction Set Extension* (*MMX*) for the next generation of Pentium processors (P55C). The SIMD concept was quickly adopted by other processor manufacturers. Motorola, for instance, developed the *AltiVec* instruction set. It has also become an integral part of new 64-bit architectures such as in *IA-64* architecture from Intel and the *x86-64* architecture from AMD.

Thus, it is evident that SIMD-processing of image data will become a standard part of future microprocessor architectures. More and more image processing tasks can be processing in real time on standard microprocessors without the need for any expensive and awkward special hardware.

1.7.4 Software and Algorithms

The rapid progress of computer hardware may distract us from the importance of software and the mathematical foundation of the basic concepts for image processing. In the early days, image processing may have been characterized more as an “art” than as a science. It was like tapping in the dark, empirically searching for a solution. Once an algorithm worked for a certain task, you could be sure that it would not work with other images and you would not even know why. Fortunately, this is gradually changing. Image processing is about to mature to a well-developed science. The deeper understanding has also led to a more realistic assessment of today’s capabilities of image processing and analysis, which in many respects is still worlds away from the capability of human vision.

It is a widespread misconception that a better mathematical foundation for image processing is of interest only to the theoreticians and has no real consequences for the applications. The contrary is true. The advantages are tremendous. In the first place, mathematical analysis allows a distinction between image processing problems that can and those that cannot be solved. This is already very helpful. Image processing algorithms become predictable and accurate, and in some cases optimal results are known. New mathematical methods often result in novel approaches that can solve previously intractable problems or that are much faster or more accurately than previous approaches. Often the speed up that can be gained by a fast algorithm is considerable. In some cases it can reach up to several orders of magnitude. Thus fast algorithms make many image processing techniques applicable and reduce the hardware costs considerably.

1.8 Further Readings‡

In this section, we give some hints on further readings in image processing.

Elementary textbooks. "The Image Processing Handbook" by Russ [158] is an excellent elementary introduction to image processing with a wealth of application examples and illustrations. Another excellent elementary textbook is Nalwa [130]. He gives — as the title indicates — a guided tour of computer vision.

Advanced textbooks. Still worthwhile to read is the classical, now almost twenty year old textbook "Digital Picture Processing" from Rosenfeld and Kak [157]. Other classical, but now somewhat outdated textbooks include Gonzalez and Woods [55], Pratt [142], and Jain [86]. The textbook of van der Heijden [188] discusses image-based measurements including parameter estimation and object recognition.

Collection of articles. An excellent overview of image processing with direct access to some key original articles is given by the following collections of articles: "Digital Image Processing" by Chelappa [19], "Readings in Computer Vision: Issues, Problems, Principles, and Paradigms" by Fischler and Firschein [41], and "Computer Vision: Principles and Advances and Applications" by Kasturi and Jain [92, 93].

Handbooks. The "Practical Handbook on Image Processing for Scientific Applications" by Jähne [81] provides a task-oriented approach with many practical procedures and tips. A state-of-the-art survey of computer vision is given by the three-volume "Handbook of Computer Vision and Applications by Jähne et al. [83]. Algorithms for image processing and computer vision are provided by Voss and Süße [192], Pitas [139], Parker [135], Umbaugh [186], and Wilson and Ritter [198].

Textbooks covering special topics. Because of the cross-disciplinary nature of image processing (Section 1.5), image processing can be treated from quite different points of view. A collection of monographs is listed here that focus on one or the other aspect of image processing:

Topic	References
Image sensors	Holst [69], Howell [74], Janesick [88]
MR imaging	Haacke et al. [59], Liang and Lauterbur [110]
Geometrical aspects of computer vision	Faugeras [37]
Perception	Mallot [117], Wandell [193]
Machine vision	Jain et al. [87], Demant et al. [27]
Robot vision	Horn [73]
Signal processing	Granlund and Knutsson [57], Lim [112]
Satellite imaging and remote sensing	Richards and Jia [152], Schott [165]
Industrial image processing	Demant et al. [27]
Object classification and pattern recognition	Schürmann [166], Bishop [9]
High-level vision	Ullman [185]

2 Image Representation

2.1 Introduction

This chapter centers around the question of how to represent the information contained in images. Together with the next two chapters it lays the mathematical foundations for low-level image processing. Two key points are emphasized in this chapter.

First, the information contained in images can be represented in entirely different ways. The most important are the spatial representation (Section 2.2) and wave number representation (Section 2.3). These representations just look at spatial data from different points of view. Since the various representations are complete and equivalent, they can be converted into each other. The conversion between the spatial and wave number representation is the well-known *Fourier transform*. This transform is an example of a more general class of operations, the *unitary transforms* (Section 2.4).

Second, we discuss how these representations can be handled with digital computers. How are images represented by arrays of digital numbers in an adequate way? How are these data handled efficiently? Can fast algorithms be devised to convert one representation into another? A key example is the fast Fourier transform, discussed in Section 2.5.

2.2 Spatial Representation of Digital Images

2.2.1 Pixel and Voxel

Images constitute a spatial distribution of the *irradiance* at a plane. Mathematically speaking, the spatial irradiance distribution can be described as a continuous function of two spatial variables:

$$E(x_1, x_2) = E(\boldsymbol{x}). \tag{2.1}$$

Computers cannot handle continuous images but only arrays of digital numbers. Thus it is required to represent images as two-dimensional arrays of points. A point on the 2-D grid is called a *pixel* or *pel*. Both words are abbreviations of the word picture element. A pixel represents the irradiance at the corresponding grid position. In the simplest case, the pixels are located on a rectangular grid. The position of the pixel

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2
Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

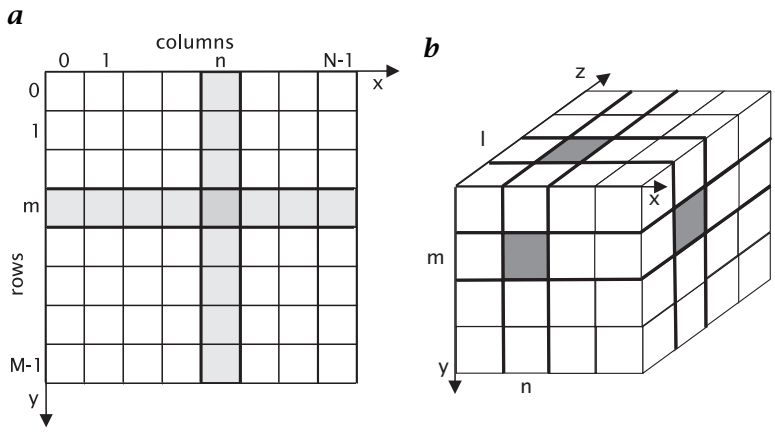

Figure 2.1: *Representation of digital images by arrays of discrete points on a rectangular grid:* ***a*** *2-D image,* ***b*** *3-D image.*

is given in the common notation for matrices. The first index, m, denotes the position of the row, the second, n, the position of the column (Fig. 2.1a). If the digital image contains $M \times N$ pixels, i. e., is represented by an $M \times N$ matrix, the index n runs from 0 to $N-1$, and the index m from 0 to $M-1$. M gives the number of rows, N the number of columns. In accordance with the matrix notation, the vertical axis (y axis) runs from top to bottom and not vice versa as it is common in graphs. The horizontal axis (x axis) runs as usual from left to right.

Each pixel represents not just a point in the image but rather a rectangular region, the elementary cell of the grid. The value associated with the pixel must represent the average irradiance in the corresponding cell in an appropriate way. Figure 2.2 shows one and the same image represented with a different number of pixels as indicated in the legend. With large pixel sizes (Fig. 2.2a, b), not only is the spatial resolution poor, but the gray value discontinuities at pixel edges appear as disturbing artifacts distracting us from the content of the image. As the pixels become smaller, the effect becomes less pronounced up to the point where we get the impression of a spatially continuous image. This happens when the pixels become smaller than the spatial resolution of our visual system. You can convince yourself of this relation by observing Fig. 2.2 from different distances.

How many pixels are sufficient? There is no general answer to this question. For visual observation of a digital image, the pixel size should be smaller than the spatial resolution of the visual system from a nominal observer distance. For a given task the pixel size should be smaller than the finest scales of the objects that we want to study. We generally find, however, that it is the available sensor technology (see Section 1.7.1)

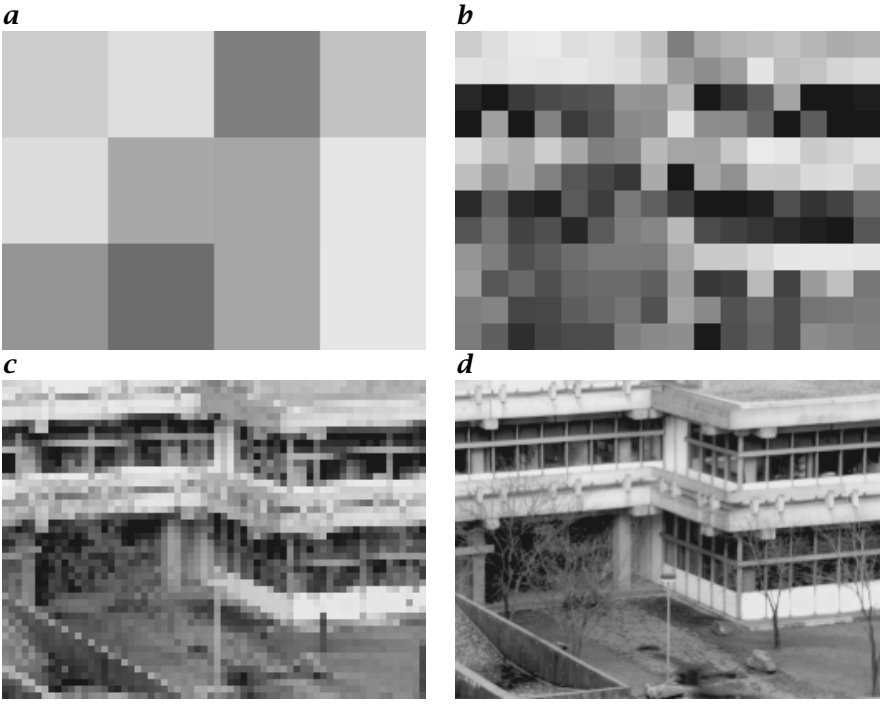

Figure 2.2: *Digital images consist of pixels. On a square grid, each pixel represents a square region of the image. The figure shows the same image with* ***a*** 3×4, ***b*** 12×16, ***c*** 48×64, *and* ***d*** 192×256 *pixels. If the image contains sufficient pixels, it appears to be continuous.*

that limits the number of pixels rather than the demands from the applications. Even a high-resolution sensor array with 1000×1000 elements has a relative spatial resolution of only 10^{-3}. This is a rather poor resolution compared to other measurements such as those of length, electrical voltage or frequency, which can be performed with relative resolutions of far beyond 10^{-6}. However, these techniques provide only a measurement at a single point, while a 1000×1000 image contains *one million* points. Thus we obtain an insight into the spatial variations of a signal. If we take image sequences, also the temporal changes and, thus, the kinematics and dynamics of the studied object become apparent. In this way, images open up a whole new world of information.

A rectangular grid is only the simplest geometry for a digital image. Other geometrical arrangements of the pixels and geometric forms of the elementary cells are possible. Finding the possible configurations is the 2-D analogue of the classification of crystal structure in 3-D space, a subject familiar to solid state physicists, mineralogists, and chemists. Crystals show periodic 3-D patterns of the arrangements of their atoms,

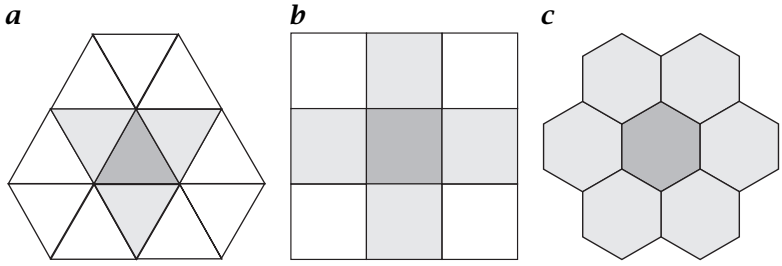

Figure 2.3: *The three possible regular grids in 2-D:* ***a*** *triangular grid,* ***b*** *square grid,* ***c*** *hexagonal grid.*

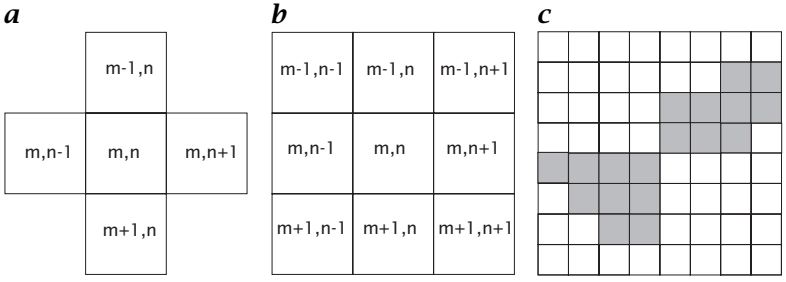

Figure 2.4: *Neighborhoods on a rectangular grid:* ***a*** *4-neighborhood and* ***b*** *8-neighborhood.* ***c*** *The black region counts as one object (connected region) in an 8-neighborhood but as two objects in a 4-neighborhood.*

ions, or molecules which can be classified by their symmetries and the geometry of the elementary cell. In 2-D, classification of digital grids is much simpler than in 3-D. If we consider only regular polygons, we have only three possibilities: triangles, squares, and hexagons (Fig. 2.3).

The 3-D spaces (and even higher-dimensional spaces) are also of interest in image processing. In three-dimensional images a pixel turns into a *voxel*, an abbreviation of *volume element*. On a rectangular grid, each voxel represents the mean gray value of a cuboid. The position of a voxel is given by three indices. The first, k, denotes the depth, m the row, and n the column (Fig. 2.1b). A Cartesian grid, i. e., hypercubic pixel, is the most general solution for digital data since it is the only geometry that can easily be extended to arbitrary dimensions.

2.2.2 Neighborhood Relations

An important property of discrete images is their *neighborhood relations* since they define what we will regard as a *connected region* and therefore as a *digital object*. A *rectangular grid* in two dimensions shows the unfortunate fact, that there are two possible ways to define neighboring pixels (Fig. 2.4a, b). We can regard pixels as neighbors either when they

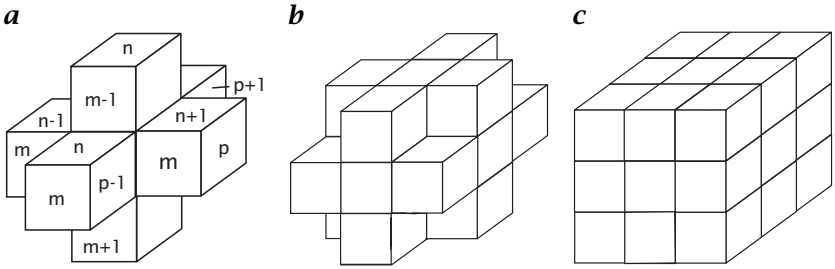

Figure 2.5: *The three types of neighborhoods on a 3-D cubic grid.* ***a*** *6-neighborhood: voxels with joint faces;* ***b*** *18-neighborhood: voxels with joint edges;* ***c*** *26-neighborhood: voxels with joint corners.*

have a joint edge or when they have at least one joint corner. Thus a pixel has four or eight neighbors and we speak of a *4-neighborhood* or an *8-neighborhood*.

Both types of neighborhood are needed for a proper definition of objects as connected regions. A region or an object is called connected when we can reach any pixel in the region by walking from one neighboring pixel to the next. The black object shown in Fig. 2.4c is one object in the 8-neighborhood, but constitutes two objects in the 4-neighborhood. The white background, however, shows the same property. Thus we have either two connected regions in the 8-neigborhood crossing each other or two separated regions in the 4-neighborhood. This inconsistency can be overcome if we declare the objects as 4-neighboring and the background as 8-neighboring, or vice versa.

These complications occur not only with a *rectangular grid*. With a *triangular grid* we can define a 3-neighborhood and a 12-neighborhood where the neighbors have either a common edge or a common corner, respectively (Fig. 2.3a). On a hexagonal grid, however, we can only define a *6-neighborhood* because pixels which have a joint corner, but no joint edge, do not exist. Neighboring pixels always have one joint edge and two joint corners. Despite this advantage, hexagonal grids are hardly used in image processing, as the imaging sensors generate pixels on a rectangular grid. The photosensors on the retina in the human eye, however, have a more hexagonal shape [193].

In three dimensions, the neighborhood relations are more complex. Now, there are three ways to define a neighbor: voxels with joint faces, joint edges, and joint corners. These definitions result in a 6-neighborhood, an 18-neighborhood, and a 26-neighborhood, respectively (Fig. 2.5). Again, we are forced to define two different neighborhoods for objects and the background in order to achieve a consistent definition of connected regions. The objects and background must be a 6-neighborhood and a 26-neighborhood, respectively, or vice versa.

2.2.3 Discrete Geometry

The discrete nature of digital images makes it necessary to redefine elementary geometrical properties such as distance, slope of a line, and coordinate transforms such as translation, rotation, and scaling. These quantities are required for the definition and measurement of geometric parameters of object in digital images.

In order to discuss the discrete geometry properly, we introduce the *grid vector* that represents the position of the pixel. The following discussion is restricted to rectangular grids. The grid vector is defined in 2-D, 3-D, and 4-D spatiotemporal images as

$$\boldsymbol{r}_{m,n} = \begin{bmatrix} n\Delta x \\ m\Delta y \end{bmatrix}, \quad \boldsymbol{r}_{l,m,n} = \begin{bmatrix} n\Delta x \\ m\Delta y \\ l\Delta z \end{bmatrix}, \quad \boldsymbol{r}_{k,l,m,n} = \begin{bmatrix} n\Delta x \\ m\Delta y \\ l\Delta z \\ k\Delta t \end{bmatrix}. \tag{2.2}$$

To measure distances, it is still possible to transfer the *Euclidian distance* from continuous space to a discrete grid with the definition

$$d_e(\boldsymbol{r},\boldsymbol{r}') = \|\boldsymbol{r}-\boldsymbol{r}'\| = \left[(n-n')^2\Delta x^2 + (m-m')^2\Delta y^2\right]^{1/2}. \tag{2.3}$$

Equivalent definitions can be given for higher dimensions. In digital images two other metrics have often been used. The *city block distance*

$$d_b(\boldsymbol{r},\boldsymbol{r}') = |n-n'| + |m-m'| \tag{2.4}$$

gives the length of a path, if we can only walk in horizontal and vertical directions (4-neighborhood). In contrast, the *chess board distance* is defined as the maximum of the horizontal and vertical distance

$$d_c(\boldsymbol{r},\boldsymbol{r}') = \max(|n-n'|, |m-m'|). \tag{2.5}$$

For practical applications, only the Euclidian distance is relevant. It is the only metric on digital images that preserves the isotropy of the continuous space. With the city block distance, for example, distances in the direction of the diagonals are longer than the Euclidean distance. The curve with equal distances to a point is not a circle but a diamond-shape curve, a square tilted by 45°.

Translation on a discrete grid is only defined in multiples of the pixel or voxel distances

$$\boldsymbol{r}'_{m,n} = \boldsymbol{r}_{m,n} + \boldsymbol{t}_{m',n'}, \tag{2.6}$$

i. e., by addition of a grid vector $\boldsymbol{t}_{m',n'}$.

Likewise, *scaling* is possible only for integer multiples of the scaling factor by taking every qth pixel on every pth line. Since this discrete scaling operation subsamples the grid, it remains to be seen whether the scaled version of the image is still a valid representation.

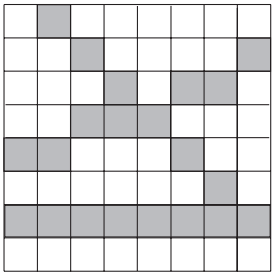

Figure 2.6: *A discrete line is only well defined in the directions of axes and diagonals. In all other directions, a line appears as a staircase-like jagged pixel sequence.*

Rotation on a discrete grid is not possible except for some trivial angles. The condition is that all points of the rotated grid coincide with the grid points. On a rectangular grid, only rotations by multiples of 180° are possible, on a square grid by multiples of 90°, and on a hexagonal grid by multiples of 60°.

Generally, the correct representation even of simple geometric objects such as lines and circles is not clear. Lines are well-defined only for angles with values of multiples of 45°, whereas for all other directions they appear as jagged, staircase-like sequences of pixels (Fig. 2.6).

All these limitations of digital geometry cause errors in the position, size, and orientation of objects. It is necessary to investigate the consequences of these errors for subsequent processing carefully.

2.2.4 Quantization

For use with a computer, the measured irradiance at the image plane must be mapped onto a limited number Q of discrete gray values. This process is called *quantization*. The number of required quantization levels in image processing can be discussed with respect to two criteria.

First, we may argue that no gray value steps should be recognized by our visual system, just as we do not see the individual pixels in digital images. Figure 2.7 shows images quantized with 2 to 16 levels of gray values. It can be seen clearly that a low number of gray values leads to false edges and makes it very difficult to recognize objects that show slow spatial variation in gray values. In printed images, 16 levels of gray values seem to be sufficient, but on a monitor we would still be able to see the gray value steps.

Generally, image data are quantized into 256 gray values. Then each pixel occupies 8 bits or one byte. This bit size is well adapted to the architecture of standard computers that can address memory bytewise. Furthermore, the resolution is good enough to give us the illusion of a

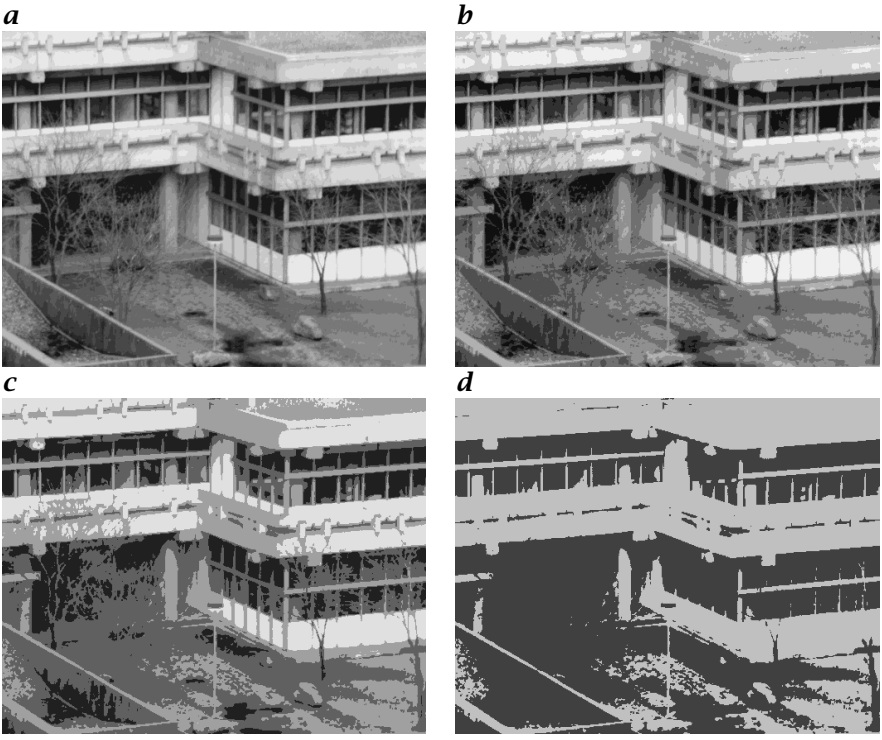

Figure 2.7: *Illustration of quantization. The same image is shown with different quantization levels:* ***a*** *16,* ***b*** *8,* ***c*** *4,* ***d*** *2. Too few quantization levels produce false edges and make features with low contrast partly or totally disappear.*

continuous change in the gray values, since the relative intensity resolution of our visual system is no better than about 2 %.

The other criterion is related to the imaging task. For a simple application in machine vision, where homogeneously illuminated objects must be detected and measured, only two quantization levels, i. e., a *binary image*, may be sufficient. Other applications such as imaging spectroscopy or medical diagnosis with x-ray images require the resolution of faint changes in intensity. Then the standard 8-bit resolution would be too coarse.

2.2.5 Signed Representation of Images‡

Normally we think of "brightness" (irradiance or radiance) as a positive quantity. Consequently, it appears natural to represent it by unsigned numbers ranging in an 8-bit representation, for example, from 0 to 255. This representation causes problems, however, as soon as we perform arithmetic operations with images. Subtracting two images is a simple example that can produce negative numbers. Since negative gray values cannot be represented, they wrap around

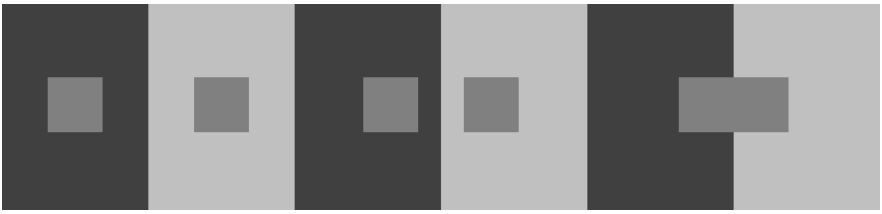

Figure 2.8: *The context determines how "bright" we perceive an object to be. Both squares have the same brightness, but the square on the dark background appears brighter than the square on the light background. The two squares only appear equally bright if they touch each other.*

and appear as large positive values. The number -1, for example, results in the positive value 255 given that -1 modulo $256 = 255$. Thus we are confronted with the problem of two different representations of gray values, as unsigned and signed 8-bit numbers. Correspondingly, we must have several versions of each algorithm, one for unsigned gray values, one for signed values, and others for mixed cases.

One solution to this problem is to handle gray values *always* as signed numbers. In an 8-bit representation, we can convert unsigned numbers into signed numbers by subtracting 128:

$$q' = (q - 128) \mod 256, \quad 0 \le q < 256. \tag{2.7}$$

Then the mean gray value intensity of 128 becomes the gray value zero and gray values lower than this mean value become negative. Essentially, we regard gray values in this representation as a deviation from a mean value.

This operation converts unsigned gray values to signed gray values which can be stored and processed as such. Only for display must we convert the gray values again to unsigned values by the inverse point operation

$$q = (q' + 128) \mod 256, \quad -128 \le q' < 128, \tag{2.8}$$

which is the same operation as in Eq. (2.7) since all calculations are performed modulo 256.

2.2.6 Luminance Perception of the Human Visual System

With respect to quantization, it is important to know how the human visual system perceives the levels and what luminance differences can be distinguished. Figure 2.8 demonstrates that the small rectangle with a medium luminance appears brighter against the dark background than against the light one, though its absolute luminance is the same. This deception only disappears when the two areas become adjacent.

The human visual system shows rather a logarithmic than a linear response. This means that we perceive relative and not absolute luminance differences equally well. In a wide range of luminance values, we can resolve relative differences of about 2%. This threshold value depends on

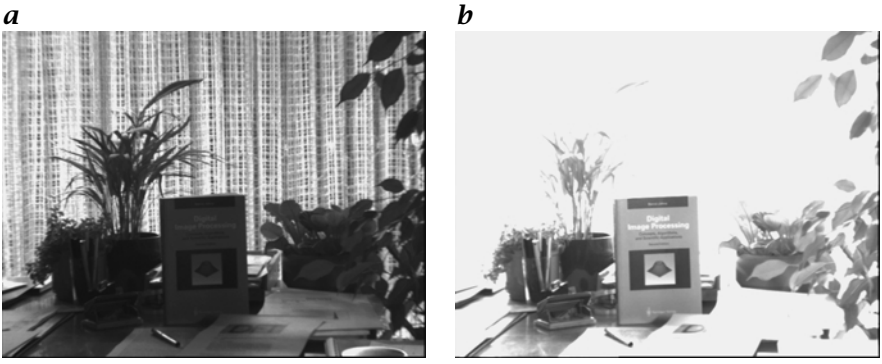

Figure 2.9: *A high-contrast scene captured by a CCD camera with a linear contrast and* ***a*** *a small and* ***b*** *a large aperture.*

a number of factors, especially the spatial frequency (wavelength) of the pattern used for the experiment. At a certain wavelength the luminance resolution is optimal.

The characteristics of the human visual system discussed above are quite different from those of a machine vision system. Typically only 256 gray values are resolved. Thus a digitized image has much lower dynamics than the human visual system. This is the reason why the quality of a digitized image, especially of a scene with high luminance contrast, appears inferior to us compared to what we see directly. In a digital image taken from such a scene with a linear image sensor, either the bright parts are overexposed or the dark parts are underexposed. This is demonstrated by the high-contrast scene in Fig. 2.9.

Although the *relative* resolution is far better than 2% in the bright parts of the image, it is poor in the dark parts. At a gray value of 10, the luminance resolution is only 10%.

One solution for coping with large dynamics in scenes is used in video cameras, which generally convert the irradiance E not linearly, but with an exponential law into the gray value g:

$$g = E^{\gamma}. \tag{2.9}$$

The exponent γ is denoted the *gamma value*. Typically, γ has a value of 0.4. With this exponential conversion, the logarithmic characteristic of the human visual system may be approximated. The contrast range is significantly enhanced. If we presume a minimum relative luminance resolution of 10% and an 8 bit gray scale range, we get contrast ranges of 25 and 316 with $\gamma = 1$ and $\gamma = 0.4$, respectively.

For many scientific applications, however, it is essential that a linear relation is maintained between the radiance of the observed object and the gray value in the digital image. Thus the gamma value must be set to one for these applications.

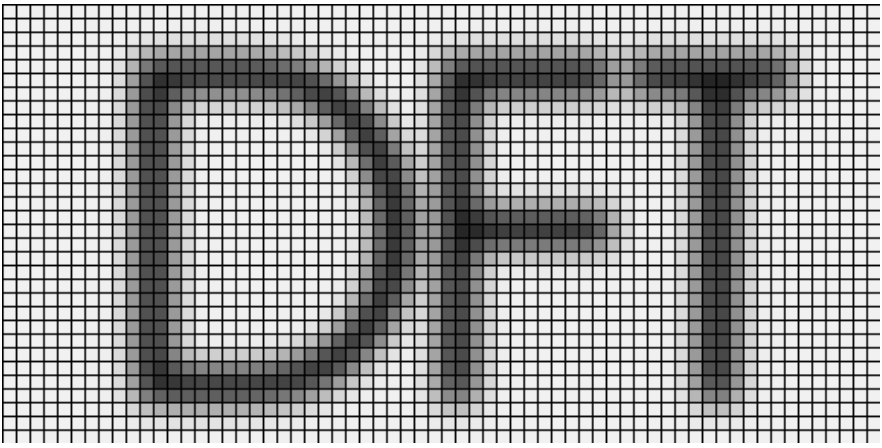

Figure 2.10: *An image can be thought to be composed of basis images in which only one pixel is unequal to zero.*

2.3 Wave Number Space and Fourier Transform

2.3.1 Vector Spaces

In Section 2.2, the discussion centered around the spatial representation of digital images. Without mentioning it explicitly, we thought of an image as composed of individual pixels (Fig. 2.10). Thus we can compose each image with basis images where just one pixel has a value of one while all other pixels are zero. We denote such a *basis image* with a one at row m, column n by

$$ {}^{m,n}\boldsymbol{P}: \quad {}^{m,n}p_{m',n'} = \begin{cases} 1 & m = m' \wedge n = n' \\ 0 & \text{otherwise.} \end{cases} \tag{2.10} $$

Any arbitrary scalar image can then be composed from the basis images in Eq. (2.10) by

$$ \boldsymbol{G} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g_{m,n}\, {}^{m,n}\boldsymbol{P}, \tag{2.11} $$

where $G_{m,n}$ denotes the gray value at the position (m,n).

It is easy to convince ourselves that the basis images ${}^{m,n}\boldsymbol{P}$ form an *orthonormal base*. To that end we require an *inner product* (also known as *scalar product*) which can be defined similarly to the scalar product for vectors. The inner product of two images $\boldsymbol{G}$ and $\boldsymbol{H}$ is defined as

$$ \langle \boldsymbol{G} | \boldsymbol{H} \rangle = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g_{m,n} h_{m,n}, \tag{2.12} $$

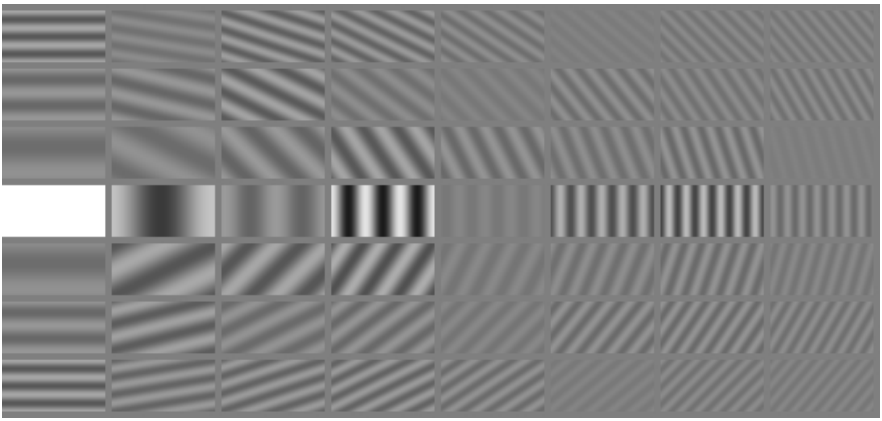

Figure 2.11: *The first 56 periodic patterns, the basis images of the Fourier transform, from which the image in Fig. 2.10 is composed.*

where the notation for the inner product from quantum mechanics is used in order to distinguish it from matrix multiplication, which is denoted by $\boldsymbol{GH}$.

From Eq. (2.12), we can immediately derive the *orthonormality relation* for the basis images ${}^{m,n}\boldsymbol{P}$:

$$\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} {}^{m',n'}p_{m,n}\,{}^{m'',n''}p_{m,n} = \delta_{m'-m''}\delta_{n'-n''}. \tag{2.13}$$

This says that the inner product between two base images is zero if two different basis images are taken. The scalar product of a basis image with itself is one. The MN basis images thus span an $M \times N$-dimensional vector space over the set of real numbers.

The analogy to the well-known two- and three-dimensional vector spaces $\mathbb{R}^2$ and $\mathbb{R}^3$ helps us to understand how other representations for images can be gained. An $M \times N$ image represents a point in the $M \times N$ vector space. If we change the coordinate system, the image remains the same but its coordinates change. This means that we just observe the same piece of information from a different point of view. We can draw two important conclusions from this elementary fact. First, all representations are equivalent to each other. Each gives a complete representation of the image. Secondly, suitable coordinate transformations lead us from one representation to the other and back again.

From the manifold of other possible representations beside the spatial representation, only one has gained prime importance for image processing. Its base images are periodic patterns and the "coordinate transform" that leads to it is known as the *Fourier transform*. Figure 2.11

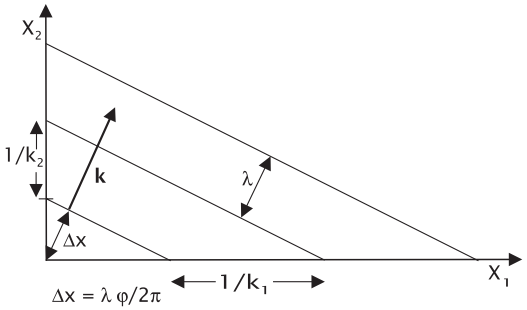

Figure 2.12: *Description of a 2-D periodic pattern by the wavelength λ, wave number* $\boldsymbol{k}$*, and phase* φ*.*

shows how the same image that has been composed by individual pixels in Fig. 2.10 is composed of periodic patterns.

A periodic pattern is first characterized by the distance between two maxima or the repetition length, the *wavelength* λ (Fig. 2.12). The direction of the pattern is best described by a vector normal to the lines of constant gray values. If we give this vector $\boldsymbol{k}$ the length $1/\lambda$

$$|\boldsymbol{k}| = 1/\lambda, \tag{2.14}$$

the wavelength and direction can be expressed by one vector, the *wave number* $\boldsymbol{k}$. The components of $\boldsymbol{k} = [k_1, k_2]^T$ directly give the number of wavelengths per unit length in the corresponding direction. The wave number $\boldsymbol{k}$ can be used for the description of periodic patterns in any dimension.

In order to complete the description of a periodic pattern, two more quantities are required: the amplitude r and the relative position of the pattern at the origin (Fig. 2.12). The position is given as the distance Δx of the first maximum from the origin. Because this distance is at most a wavelength, it is best given as a *phase angle* $\varphi = 2\pi\Delta x/\lambda = 2\pi k \cdot \Delta x$ (Fig. 2.12) and the complete description of a periodic pattern is given by

$$r\cos(2\pi\boldsymbol{k}^T\boldsymbol{x} - \varphi). \tag{2.15}$$

This description is, however, mathematically quite awkward. We rather want a simple factor by which the base patterns have to be multiplied, in order to achieve a simple decomposition in periodic patterns. This is only possible by using *complex numbers* $\hat{g} = r\exp(-\mathrm{i}\varphi)$ and the complex exponential function $\exp(\mathrm{i}\varphi) = \cos\varphi + \mathrm{i}\sin\varphi$. The real part of $\hat{g}\exp(2\pi\mathrm{i}\boldsymbol{k}^T\boldsymbol{x})$ gives the periodic pattern in Eq. (2.15):

$$\Re(\hat{g}\exp(2\pi\mathrm{i}\boldsymbol{k}^T\boldsymbol{x})) = r\cos(2\pi\boldsymbol{k}^T\boldsymbol{x} - \varphi). \tag{2.16}$$

In this way the decomposition into periodic patterns requires the extension of real numbers to complex numbers. A real-valued image is thus considered as a complex-valued image with a zero imaginary part.

The subject of the remainder of this chapter is rather mathematical, but it forms the base for image representation and low-level image processing. After introducing both the continuous and discrete Fourier transform in Sections 2.3.2 and 2.3.3, we will discuss all properties of the Fourier transform that are of relevance to image processing in Section 2.3.5. We will take advantage of the fact that we are dealing with images, which makes it easy to illustrate some complex mathematical relations.

2.3.2 One-Dimensional Fourier Transform

First, we will consider the *one-dimensional Fourier transform*.

Definition 1 (1-D FT) *]If $g(x) : \mathbb{R} \mapsto \mathbb{C}$ is a square integrable function, that is,*

$$\int_{-\infty}^{\infty} |g(x)|^2 \, \mathrm{d}x < \infty, \tag{2.17}$$

then the Fourier transform *of $g(x)$, $\hat{g}(k)$ is given by*

$$\hat{g}(k) = \int_{-\infty}^{\infty} g(x) \exp(-2\pi \mathrm{i} k x) \, \mathrm{d}x. \tag{2.18}$$

The Fourier transform maps the vector space of square integrable functions onto itself. The inverse Fourier transform *of $\hat{g}(k)$ results in the original function $g(x)$:*

$$g(x) = \int_{-\infty}^{\infty} \hat{g}(k) \exp(2\pi \mathrm{i} k x) \, \mathrm{d}k. \tag{2.19}$$

The Fourier transform can be written in a more compact form if the abbreviation

$$\mathrm{w} = \mathrm{e}^{2\pi \mathrm{i}} \tag{2.20}$$

is used and the integral is written as an *inner product*:

$$\langle g(x) | h(x) \rangle = \int_{-\infty}^{\infty} g^*(x) h(x) \mathrm{d}x, \tag{2.21}$$

where * denotes the conjugate complex. Then

$$\hat{g}(k) = \left\langle \mathrm{w}^{kx} \, | \, g(x) \right\rangle. \tag{2.22}$$

The function w^t can be visualized as a vector that rotates anticlockwise on the *unit circle* in the *complex plane*. The variable t gives the number of revolutions.

Sometimes, it is also convenient to use an operator notation for the Fourier transform:

$$\hat{g} = \mathcal{F} g \quad \text{and} \quad g = \mathcal{F}^{-1} \hat{g}. \tag{2.23}$$

A function and its transform, a *Fourier transform pair*, is simply denoted by $g(x) \circ\!\!-\!\!\bullet\, \hat{g}(k)$.

For the *discrete Fourier transform* (*DFT*), the wave number is now an integer number that indicates how many wavelengths fit into the interval with N elements.

Definition 2 (1-D DFT) *The DFT maps an ordered N-tuple of complex numbers g_n, the* complex-valued vector $\boldsymbol{g}$*,*

$$\boldsymbol{g} = [g_0, g_1, \ldots, g_{N-1}]^T, \tag{2.24}$$

onto another vector $\hat{\boldsymbol{g}}$ of a vector space with the same dimension N.

$$\hat{g}_v = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} g_n \exp\left(-\frac{2\pi \mathrm{i} n v}{N}\right), \quad 0 \le v < N. \tag{2.25}$$

The back transformation is given by

$$g_n = \frac{1}{\sqrt{N}} \sum_{v=0}^{N-1} \hat{g}_v \exp\left(\frac{2\pi \mathrm{i} n v}{N}\right), \quad 0 \le n < N. \tag{2.26}$$

Again it is useful to use a convenient abbreviation for the kernel of the DFT; compare Eq. (2.20):

$$\mathrm{w}_N = \mathrm{w}^{1/N} = \exp\left(\frac{2\pi \mathrm{i}}{N}\right) \tag{2.27}$$

As the continuous Fourier transform, the DFT can be considered as the inner product of the vector $\boldsymbol{g}$ with a set of N orthonormal basis vectors

$$\boldsymbol{b}_v = \frac{1}{\sqrt{N}} \left[\mathrm{w}_N^0, \mathrm{w}_N^v, \mathrm{w}_N^{2v}, \ldots, \mathrm{w}_N^{(N-1)v}\right]^T. \tag{2.28}$$

Then

$$\hat{g}_v = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \mathrm{w}_N^{-nv} g_n = \langle \boldsymbol{b}_v \,|\, \boldsymbol{g} \rangle = \boldsymbol{b}_v{}^T \boldsymbol{g}. \tag{2.29}$$

Note the second compact notation of the scalar product between two vectors on the right-hand side of the equation that will be used in the following.

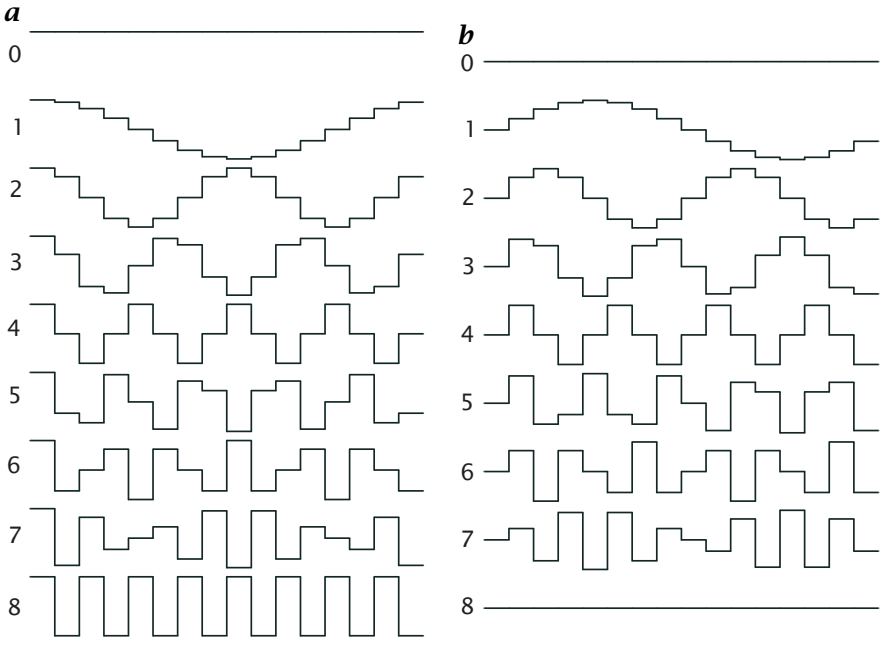

Figure 2.13: *The first 9 basis functions of the DFT for $N = 16$; **a** real part (cosine function), **b** imaginary part (sine function).*

Equation (2.29) means that the coefficient $\hat{g}_v$ in the Fourier space is obtained by projecting the vector $\boldsymbol{g}$ onto the basis vector $\boldsymbol{b}_v$. The N basis vectors $\boldsymbol{b}_v$ are orthogonal to each other:

$$\boldsymbol{b}_v{}^T \boldsymbol{b}_{v'} = \delta_{v-v'} = \begin{cases} 1 & v = v' \\ 0 & \text{otherwise.} \end{cases} \tag{2.30}$$

Consequently, the set $\boldsymbol{b}_v$ forms an orthonormal basis for the *vector space*, which means that each vector of the vector space can be expressed as a linear combination of the basis vectors of the Fourier space. The DFT calculates the projections of the vector $\boldsymbol{g}$ onto all the basis vectors directly, i. e., the components of $\boldsymbol{g}$ in the direction of the basis vectors. In this sense, the DFT is just a special type of coordinate transformation in an M-dimensional vector space. Mathematically, the DFT differs from more familiar coordinate transformations such as rotation in a three-dimensional vector space (Section 7.2.2) only because the vector space is over the field of the complex instead of real numbers and has many more dimensions.

The real and imaginary parts of the basis vectors are sampled sine and cosine functions of different wavelengths (Fig. 2.13). The index v denotes how often the wavelength of the function fits into the interval

Table 2.1: *Comparison of the continuous Fourier transform (FT), the Fourier series (FS), the infinite discrete Fourier transform (IDFT), and the discrete Fourier transform (DFT) in one dimension* ($\mathrm{w} = \mathrm{e}^{2\pi\mathrm{i}}$).

Type	Forward transform	Backward transform
FT: $\mathbb{R} \mapsto \mathbb{R}$	$\hat{g}(k) = \int_{-\infty}^{\infty} g(x)\mathrm{w}^{-kx}\mathrm{d}x$	$g(x) = \int_{-\infty}^{\infty} \hat{g}(k)\mathrm{w}^{kx}\mathrm{d}k$
FS: $[0, \Delta x] \mapsto \mathbb{Z}$	$\hat{g}_v = \frac{1}{\Delta x}\int_0^{\Delta x} g(x)\mathrm{w}^{-vx/\Delta x}\mathrm{d}x$	$g(x) = \sum_{v=-\infty}^{\infty} \hat{g}_v \mathrm{w}^{vx/\Delta x}$
IDFT: $\mathbb{Z} \mapsto [0, 1/\Delta x]$	$\hat{g}(k) = \sum_{n=-\infty}^{\infty} g_n \mathrm{w}^{-nk\Delta x}$	$g_n = \Delta x \int_0^{1/\Delta x} \hat{g}(k)\mathrm{w}^{nk\Delta x}\mathrm{d}k$
DFT: $\mathbb{Z}_N \mapsto \mathbb{Z}_N$	$\hat{g}_v = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} g_n \mathrm{w}_N^{-vn}$	$g_n = \frac{1}{\sqrt{N}}\sum_{v=0}^{N-1} \hat{g}_v \mathrm{w}_N^{vn}$

$[0, M]$. The basis vector $\boldsymbol{b}_0$ is a constant real vector. The projection onto this vector results in the mean of the elements of the vector $\boldsymbol{g}$.

Besides the continuous and discrete Fourier transforms there are two other forms you may be familiar with: the *Fourier series* (*FS*) that maps a function in a finite interval $[0, \Delta x]$ to an infinite series of coefficients and the *infinite discrete Fourier transform* (*IDFT*) that maps an infinite series of complex numbers to a finite interval $[0, 1/\Delta x]$ in the Fourier space. Therefore it is illustrative to compare the DFT with these transforms (Table 2.1).

2.3.3 Multidimensional Fourier transform

The Fourier transform can easily be extended to *multidimensional* signals.

Definition 3 (Multidimensional FT) *If* $g(\boldsymbol{x}) : \mathbb{R}^W \mapsto \mathbb{C}$ *is a square integrable function, that is,*

$$\int_{-\infty}^{\infty} |g(\boldsymbol{x})|^2 \mathrm{d}^W x = \langle g(\boldsymbol{x}) \,|\, g(\boldsymbol{x}) \rangle = ||g(\boldsymbol{x})||_2^2 < \infty \tag{2.31}$$

then the Fourier transform *of* $g(\boldsymbol{x})$, $\hat{g}(\boldsymbol{k})$ *is given by*

$$\hat{g}(\boldsymbol{k}) = \int_{-\infty}^{\infty} g(\boldsymbol{x}) \exp\left(-2\pi\mathrm{i}\boldsymbol{k}^T\boldsymbol{x}\right) \mathrm{d}^W x = \left\langle \mathrm{w}^{\boldsymbol{x}^T\boldsymbol{k}} \,|\, g(\boldsymbol{x}) \right\rangle \tag{2.32}$$

and the inverse Fourier transform *by*

$$g(\boldsymbol{x}) = \int_{-\infty}^{\infty} \hat{g}(\boldsymbol{k}) \exp\left(2\pi \mathrm{i} \boldsymbol{k}^T \boldsymbol{x}\right) \mathrm{d}^W k = \left\langle \mathrm{w}^{-\boldsymbol{x}^T \boldsymbol{k}} \,|\, \hat{g}(\boldsymbol{k}) \right\rangle . \tag{2.33}$$

The scalar product in the exponent of the kernel $\boldsymbol{x}^T\boldsymbol{k}$ makes the kernel of the Fourier transform separable, that is, it can be written as

$$\mathrm{w}^{\boldsymbol{x}^T\boldsymbol{k}} = \prod_{p=1}^{W} \mathrm{w}^{k_p x_p}. \tag{2.34}$$

The discrete Fourier transform is discussed here for two dimensions. The extension to higher dimensions is straightforward.

Definition 4 (2-D DFT) *The 2-D DFT maps an $M \times N$ complex-valued matrices on $M \times N$ complex-valued matrices:*

$$\begin{aligned} \hat{g}_{u,v} &= \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g_{m,n} \exp\left(-\frac{2\pi \mathrm{i} m u}{M}\right) \exp\left(-\frac{2\pi \mathrm{i} n v}{N}\right) \\ &= \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \left(\sum_{n=0}^{N-1} g_{m,n} \mathrm{w}_N^{-nv} \right) \mathrm{w}_M^{-mu}. \end{aligned} \tag{2.35}$$

In the second line, the abbreviation defined in Eq. (2.27) is used. As in the one-dimensional case, the DFT expands a matrix into a set of NM basis matrices which spans the $N \times M$-dimensional vector space over the field of complex numbers. The basis matrices are of the form

$$\underbrace{\boldsymbol{B}_{u,v}}_{M\times N} = \frac{1}{\sqrt{MN}} \begin{bmatrix} \mathrm{w}^0 \\ \mathrm{w}_M^u \\ \mathrm{w}_M^{2u} \\ \vdots \\ \mathrm{w}_M^{(M-1)u} \end{bmatrix} \left[\mathrm{w}^0, \mathrm{w}_N^v, \mathrm{w}_N^{2v}, \ldots, \mathrm{w}_N^{(N-1)v} \right]. \tag{2.36}$$

In this equation, the basis matrices are expressed as an *outer product* of the column and the row vector that form the basis vectors of the one-dimensional DFT (Eq. (2.28)). This reflects the separability of the kernel of the 2-D DFT.

Then the 2-D DFT can be written again as an inner product

$$\hat{g}_{u,v} = \langle \boldsymbol{B}_{u,v} \,|\, \boldsymbol{G} \rangle , \tag{2.37}$$

where the inner product of two complex-valued matrices is given by

$$\langle \boldsymbol{G} \,|\, \boldsymbol{H} \rangle = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g_{m,n}^* h_{m,n}. \tag{2.38}$$

The inverse 2-D DFT is given by

$$g_{mn} = \frac{1}{\sqrt{MN}} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \hat{g}_{u,v} \mathrm{w}_M^{mu} \mathrm{w}_N^{nv} = \left\langle \boldsymbol{B}_{-m,-n} \,\middle|\, \hat{\boldsymbol{G}} \right\rangle . \tag{2.39}$$

2.3.4 Alternative Definitions‡

In the literature several variations of the Fourier transform exist, which can lead to a lot of confusions and errors. This has to do with the definition of the wave number. The definition of the wave number as a reciprocal wavelength $k = 1/\lambda$ is the most useful for signal processing, because k directly gives the number of wavelengths per unit length. In physics and electrical engineering, however, a definition including the factor 2π is more common: $\check{k} = 2\pi/\lambda$. With this notation, two forms of the Fourier transform can be defined: the asymmetric form

$$\hat{g}(\check{k}) = \left\langle \exp(\mathrm{i}\check{k}x) \,|\, g(x) \right\rangle , \; g(x) = \frac{1}{2\pi} \left\langle \exp(-\mathrm{i}\check{k}x) \,\middle|\, \hat{g}(\check{k}) \right\rangle \tag{2.40}$$

and the symmetric form

$$\hat{g}(\check{k}) = \frac{1}{\sqrt{2\pi}} \left\langle \exp(\mathrm{i}\check{k}x) \,|\, g(x) \right\rangle , \; g(x) = \frac{1}{\sqrt{2\pi}} \left\langle \exp(-\mathrm{i}\check{k}x) \,\middle|\, \hat{g}(\check{k}) \right\rangle . \tag{2.41}$$

Because all three versions of the Fourier transform are in common use, it is likely that wrong factors in Fourier transform pairs will be obtained. The rules for conversion of Fourier transform pairs between the three versions can directly be inferred from the definitions and are summarized here:

$$\begin{array}{lcl} k = 1/\lambda, \text{ Eq. (2.22)} & g(x) & \circ\!\!-\!\!\bullet \; \hat{g}(k) \\ \check{k} \text{ with } 2\pi, \text{ Eq. (2.40)} & g(x) & \circ\!\!-\!\!\bullet \; \hat{g}(\check{k}/2\pi) \\ \check{k} \text{ with } 2\pi, \text{ Eq. (2.41)} & g(x) & \circ\!\!-\!\!\bullet \; \hat{g}(\check{k}/\sqrt{2\pi})/\sqrt{2\pi}. \end{array} \tag{2.42}$$

2.3.5 Properties of the Fourier transform

In this section we discuss the basic properties of the continuous and discrete Fourier transform. We point out those properties of the FT that are most significant for image processing. Together with some basic Fourier transform pairs (≻ R5), these general properties (≻ R4, ≻ R7) form a powerful framework with which further properties of the Fourier transform and the transforms of many functions can be derived without much effort.

Periodicity of DFT. The kernel of the DFT Eq. (2.25) shows a characteristic *periodicity*

$$\exp\left(-\frac{2\pi\mathrm{i}(n+lN)}{N}\right) = \exp\left(-\frac{2\pi\mathrm{i}n}{N}\right), \mathrm{w}_N^{(n+lN)} = \mathrm{w}_N^n, \quad \forall\, l \in \mathbb{Z}. \tag{2.43}$$

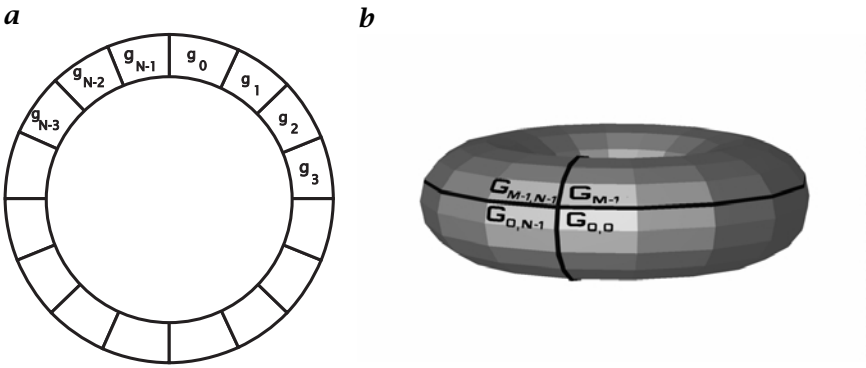

Figure 2.14: *Geometric interpretation of the* periodicity *of the one- and two-dimensional DFT with* ***a*** *the Fourier ring and* ***b*** *the Fourier torus.*

The definitions of the DFT restrict the spatial domain and the Fourier domain to a finite number of values. If we do not care about this restriction and calculate the forward and back transformation for all integer numbers, we find from Eqs. (2.37) and (2.39) the same periodicities for functions in the space and Fourier domain:

$$\begin{array}{ll} \text{wave number domain} & \hat{g}_{u+kM,v+lN} = \hat{g}_{u,v}, \quad \forall\, k,l \in \mathbb{Z} \\ \text{space domain} & g_{m+kM,n+lN} = g_{m,n}, \quad \forall\, k,l \in \mathbb{Z}. \end{array} \tag{2.44}$$

These equations state a periodic replication in all directions in both domains beyond the original definition range. The periodicity of the DFT gives rise to an interesting geometric interpretation. In the one-dimensional case, the border points g_{N-1} and $g_N = g_0$ are neighboring points. We can interpret this property geometrically by drawing the points of the vector not on a finite line but on a circle, the so-called *Fourier ring* (Fig. 2.14a). This representation has a deeper meaning when we consider the Fourier transform as a special case of the *z-transform* [133]. With two dimensions, a matrix is mapped onto a torus (Fig. 2.14b), the *Fourier torus*.

Symmetries. Four types of *symmetries* are important for the Fourier transform:

$$\begin{array}{lrcl} \text{even} & g(-\boldsymbol{x}) & = & g(\boldsymbol{x}), \\ \text{odd} & g(-\boldsymbol{x}) & = & -g(\boldsymbol{x}), \\ \text{hermitian} & g(-\boldsymbol{x}) & = & g^*(\boldsymbol{x}), \\ \text{anti-hermitian} & g(-\boldsymbol{x}) & = & -g^*(\boldsymbol{x}) \end{array} \tag{2.45}$$

The symbol $*$ denotes the complex conjugate. The hermitity is of importance because the kernels of the FT Eq. (2.18) and DFT Eq. (2.25) are hermitian.

Any function $g(\boldsymbol{x})$ can be split into its even and odd parts by

$$ {}^{e}g(\boldsymbol{x}) = \frac{g(\boldsymbol{x}) + g(-\boldsymbol{x})}{2} \quad \text{and} \quad {}^{o}g(\boldsymbol{x}) = \frac{g(\boldsymbol{x}) - g(-\boldsymbol{x})}{2}. \tag{2.46} $$

With this partition, the Fourier transform can be parted into a cosine and a sine transform:

$$ \hat{g}(\boldsymbol{k}) = 2\int_0^\infty {}^{e}g(\boldsymbol{x}) \cos(2\pi \boldsymbol{k}^T\boldsymbol{x}) \mathrm{d}^W x + 2\mathrm{i}\int_0^\infty {}^{o}g(\boldsymbol{x}) \sin(2\pi \boldsymbol{k}^T\boldsymbol{x}) \mathrm{d}^W x. \tag{2.47} $$

It follows that if a function is even or odd, its transform is also even or odd. The full symmetry relations are:

real	⟷	Hermitian
imaginary	⟷	anti-Hermitian
Hermitian	⟷	real
anti-Hermitian	⟷	imaginary
even	⟷	even
odd	⟷	odd
real and even	⟷	real and even
real and odd	⟷	imaginary and odd
imaginary and even	⟷	imaginary and even
imaginary and odd	⟷	real and odd

(2.48)

The DFT shows the same symmetries as the FT (Eqs. (2.45) and (2.48)). In the definition for even and odd functions $g(-\boldsymbol{x}) = \pm g(\boldsymbol{x})$ only $\boldsymbol{x}$ must be replaced by the corresponding indices: $g_{-n} = \pm g_n$ or $g_{-m,-n} = \pm g_{m,n}$. Note that because of the periodicity of the DFT, these symmetry relations can also be written as

$$ g_{-m,-n} = \pm g_{m,n} \equiv g_{M-m,N-n} = \pm g_{m,n} \tag{2.49} $$

for even (+ sign) and odd (− sign) functions. This is equivalent to shifting the symmetry center from the origin to the point $[M/2, N/2]^T$.

The study of symmetries is important for practical purposes. Careful consideration of symmetry allows storage space to be saved and algorithms to speed up. Such a case is real-valued images. Real-valued images can be stored in half of the space as complex-valued images. From the symmetry relation Eq. (2.48) we can conclude that real-valued functions exhibit a Hermitian DFT:

$$ \begin{aligned} g_n = g_n^* \quad &\longleftrightarrow \quad \hat{g}_{N-v} = \hat{g}_v^* \\ g_{mn} = g_{mn}^* \quad &\longleftrightarrow \quad \hat{g}_{M-u,N-v} = \hat{g}_{uv}^* \end{aligned} \tag{2.50} $$

The complex-valued DFT of real-valued vectors is, therefore, completely determined by the values in one half-space. The other half-space is obtained by mirroring at the symmetry center $[N/2]$. Consequently, we

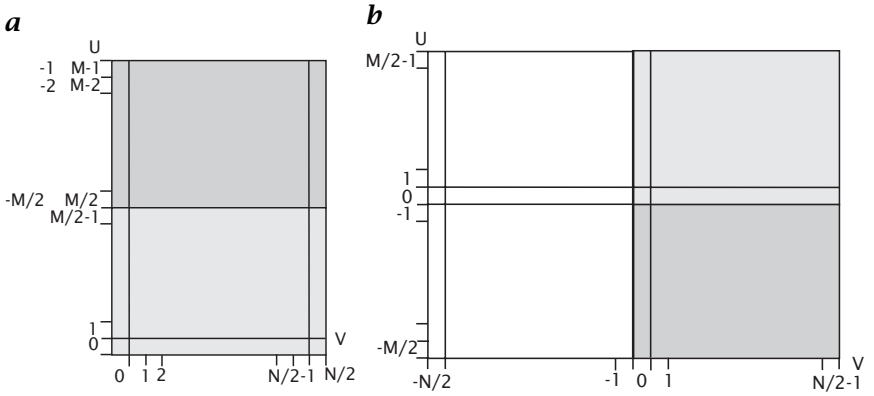

Figure 2.15: ***a*** *Half-space as computed by an in-place Fourier transform algorithm; the wave number zero is in the upper left corner;* ***b*** *FT with the missing half appended and remapped so that the wave number zero is in the center.*

need the same amount of storage space for the DFT of a real vector as for the vector itself, as only half of the complex spectrum needs to be stored.

In two and higher dimensions, matters are slightly more complex. Again, the Fourier transform of a real-valued image is determined completely by the values in one half-space, but there are many ways to select the half-space. This means that all except for one component of the wave number can be negative.

The Fourier transform of a real $M \times N$ image can be represented by M rows and $N/2+1$ columns (Fig. 2.15) assuming that N is even. Unfortunately, $N/2+1$ columns are required, because the first ($m = 0$) and last column ($m = M/2$) are symmetric to themselves according to Eq. (2.50). Thus it appears impossible to overwrite a real image by its complex Fourier transform because we need one more column. A closer examination shows that it works nevertheless. The first and last columns are real-valued because of symmetry reasons ($\hat{g}_{0,N-v} = \hat{g}^*_{0,v}$ and $\hat{g}_{M/2,N-v} = \hat{g}^*_{M/2,v}$). Therefore, the real part of column $M/2$ can be stored in the imaginary part of column 0.

For real-valued image sequences, again we need only a half-space to represent the spectrum. Physically, it makes the most sense to choose the half-space that contains positive frequencies. In contrast to a single image, we obtain the full wave-number space. Now we can identify the spatially identical wave numbers $\boldsymbol{k}$ and $-\boldsymbol{k}$ as structures propagating in opposite directions.

Separability. The kernel of the Fourier transform is separable (Eq. (2.34)). Therefore, the transform of a separable function is also separable:

$$\prod_{p=1}^{W} g(x_p) \circ\!\!-\!\!\bullet \prod_{p=1}^{W} \hat{g}(k_p). \tag{2.51}$$

This property is essential to compute transforms of multidimensional functions efficiently from 1-D transforms because many of them are separable.

Similarity. The similarity theorem states how a Fourier transform is influenced by scaling of the coordinate system. In one dimension, a function can only be scaled ($x' = ax$). In multiple dimensions, the coordinate system can be transformed in a more general way by an affine transform ($\boldsymbol{x}' = \boldsymbol{A}\boldsymbol{x}$), i. e., the new basis vectors are linear combinations of the old basis vectors. A special case is the rotation of the coordinate system.

Theorem 1 (Similarity) *Be a a non-zero real number, $\boldsymbol{A}$ a real, invertible matrix, and $\boldsymbol{R}$ an orthogonal matrix representing a rotation of the coordinate system ($\boldsymbol{R}^{-1} = \boldsymbol{R}^T$, $\det \boldsymbol{R} = 1$). Then the following similarity relations hold:*

$$\begin{array}{lcl} \textit{Scalar} & g(a\boldsymbol{x}) \circ\!\!-\!\!\bullet & \frac{1}{|a|}\hat{g}(\boldsymbol{k}/a) \\ \textit{Affine transform} & g(\boldsymbol{A}\boldsymbol{x}) \circ\!\!-\!\!\bullet & \frac{1}{\det \boldsymbol{A}}\hat{g}((\boldsymbol{A}^T)^{-1}\boldsymbol{k}) \\ \textit{Rotation} & g(\boldsymbol{R}\boldsymbol{x}) \circ\!\!-\!\!\bullet & \hat{g}(\boldsymbol{R}\boldsymbol{k}) \end{array} \tag{2.52}$$

If a function is squeezed in the spatial domain, it is stretched in the Fourier domain, and vice versa. A rotation of the coordinate system in the spatial domain causes the identical rotation in the Fourier domain.

The above similarity theorems do not apply to the discrete Fourier transform because an arbitrary scaling and rotation is not possible. A stretching of a discrete function is only possible by an integer factor K (*upsampling*) and the newly generated discrete points are filled with zeros:

$$(\boldsymbol{g}_{\uparrow K})_n = \begin{cases} g_{n/K} & n = 0, K, 2K, \ldots (N-1)K \\ 0 & \text{otherwise.} \end{cases} \tag{2.53}$$

Theorem 2 (Similarity, discrete) *Be $\boldsymbol{g}$ a complex-valued vector with N elements and $K \in \mathbb{N}$. Then the discrete Fourier transform of the upsampled vector $\boldsymbol{g}_{\uparrow K}$ with KN elements is given by*

$$\boldsymbol{g}_{\uparrow K} \circ\!\!-\!\!\bullet \frac{1}{K}\hat{\boldsymbol{g}} \quad \textit{with} \quad \hat{g}_{kN+v} = \hat{g}_v. \tag{2.54}$$

Upsampling by a factor K thus simply results in a K-fold replication of the Fourier transform. Note that because of the periodicity of the discrete Fourier transform discussed at the beginning of this section, $\hat{g}_{kN+v} = \hat{g}_v$.

Shift. In Section 2.3.1 we discussed some properties of the basis images of the Fourier space, the complex exponential functions $\exp\left(2\pi\mathrm{i}\boldsymbol{k}^T\boldsymbol{x}\right)$. A spatial shift of these functions causes a multiplication by a phase factor:

$$\exp\left(2\pi\mathrm{i}(\boldsymbol{x}-\boldsymbol{x}_0)^T\boldsymbol{k}\right) = \exp\left(-2\pi\mathrm{i}\boldsymbol{x}_0^T\boldsymbol{k}\right)\exp\left(2\pi\mathrm{i}\boldsymbol{k}^T\boldsymbol{x}\right) \tag{2.55}$$

As a direct consequence of the linearity of the Fourier transform, we can formulate the following *shift theorem*:

Theorem 3 (Shift) *If the function $g(\boldsymbol{x})$ has the Fourier transform $\hat{g}(\boldsymbol{k})$, then $g(\boldsymbol{x}-\boldsymbol{x}_0)$ has the Fourier transforms $\exp(-2\pi\mathrm{i}\boldsymbol{x}_0^T\boldsymbol{k})\hat{g}(\boldsymbol{k})$.*

Thus, a shift in the spatial domain does not change the Fourier transform except for a wave number-dependent phase change $-2\pi\boldsymbol{x}_0^T\boldsymbol{k}$.

The shift theorem can also be applied in the Fourier domain. A shift in the Fourier space, $\hat{g}(\boldsymbol{k}-\boldsymbol{k}_0)$, results in a signal in the spatial domain that is modulated by a complex exponential with the wave number vector $\boldsymbol{k}_0$: $\exp(2\pi\mathrm{i}\boldsymbol{k}_0^T\boldsymbol{x})g(\boldsymbol{x})$.

Convolution. *Convolution* is one of the most important operations for signal processing. For a continuous signal it is defined by

$$(g*h)(\boldsymbol{x}) = \int_{-\infty}^{\infty} h(\boldsymbol{x}')g(\boldsymbol{x}-\boldsymbol{x}')\mathrm{d}^W x'. \tag{2.56}$$

In signal processing, the function $h(\boldsymbol{x})$ is normally zero except for a small area around zero and is often denoted as the *convolution mask*. Thus, the convolution with $h(\boldsymbol{x})$ results in a new function $g'(\boldsymbol{x})$ whose values are a kind of weighted average of $g(\boldsymbol{x})$ in a small neighborhood around $\boldsymbol{x}$. It changes the signal in a defined way, for example, makes it smoother. Therefore it is also called a *filter*.

One- and two-dimensional discrete convolution are defined analogous to Eq. (2.56) by

$$g'_n = \sum_{n'=0}^{N-1} h_{n'}g_{n-n'}, \quad g'_{m,n} = \sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1} h_{m'n'}g_{m-m',n-n'} \tag{2.57}$$

The *convolution theorem* for the FT and DFT states:

Theorem 4 (Convolution) *If $g(\boldsymbol{x})$ $(\boldsymbol{g},\boldsymbol{G})$ has the Fourier transforms $\hat{g}(\boldsymbol{k})$ $(\hat{\boldsymbol{g}},\hat{\boldsymbol{G}})$ and $h(\boldsymbol{x})$, $(\boldsymbol{h},\boldsymbol{H})$ have the Fourier transforms $\hat{h}(\boldsymbol{k})$ $(\hat{\boldsymbol{h}},\hat{\boldsymbol{H}})$, then $h*$*

$g(\boldsymbol{h} * \boldsymbol{g}, \boldsymbol{H} * \boldsymbol{G})$ has the Fourier transforms $\hat{h}(\boldsymbol{k})\hat{g}(\boldsymbol{k})$, $(N\hat{\boldsymbol{h}} \cdot \hat{\boldsymbol{g}}, MN\hat{\boldsymbol{H}} \cdot \hat{\boldsymbol{G}})$:

$$\begin{array}{lcl} \textit{FT:} & h(\boldsymbol{x}) * g(\boldsymbol{x}) \quad \circ\!\!-\!\!\bullet & \hat{h}(\boldsymbol{k}) \cdot \hat{g}(\boldsymbol{k}) \\ \textit{1-D DFT:} & \boldsymbol{h} * \boldsymbol{g} \quad \circ\!\!-\!\!\bullet & N\hat{\boldsymbol{h}} \cdot \hat{\boldsymbol{g}} \\ \textit{2-D DFT:} & \boldsymbol{H} * \boldsymbol{G} \quad \circ\!\!-\!\!\bullet & MN\hat{\boldsymbol{H}} \cdot \hat{\boldsymbol{G}} \end{array} \tag{2.58}$$

Thus, convolution of two functions means multiplication of their transforms. Likewise, convolution of two functions in the Fourier domain means multiplication in the space domain. The simplicity of convolution in the Fourier space stems from the fact that the base functions of the Fourier domain, the complex exponentials $\exp\left(2\pi \mathrm{i} \boldsymbol{k}^T \boldsymbol{x}\right)$, are joint eigenfunctions of all convolution operators. This means that these functions are not changed by a convolution operator except for the multiplication by a factor.

From the convolution theorem, the following properties are immediately evident. Convolution is

$$\begin{array}{ll} \text{commutative} & h * g = g * h, \\ \text{associative} & h_1 * (h_2 * g) = (h_1 * h_2) * g, \\ \text{distributive over addition} & (h_1 + h_2) * g = h_1 * g + h_2 * g. \end{array} \tag{2.59}$$

In order to grasp the importance of these properties of convolution, we note that two operations that do not look so at first glance, are also convolution operations: the shift operation and all derivative operators.

In both cases the Fourier transform is only multiplied by a complex factor. For a shift operation this can be seen directly from the shift theorem (Theorem 3). The convolution mask of a shift operator S is a shifted δ distribution:

$$S(\boldsymbol{s})g(\boldsymbol{x}) = \delta(\boldsymbol{x} - \boldsymbol{s}) * g(\boldsymbol{x}). \tag{2.60}$$

For a partial derivative of a function in the spatial domain the *derivation theorem* states:

Theorem 5 (Derivation) *If $g(\boldsymbol{x})$ is differentiable for all $\boldsymbol{x}$ and has the Fourier transform $\hat{g}(\boldsymbol{k})$, then the Fourier transform of the partial derivative $\partial g(\boldsymbol{x})/\partial x_p$ is $2\pi \mathrm{i} k_p \hat{g}(\boldsymbol{k})$:*

$$\frac{\partial g(\boldsymbol{x})}{\partial x_p} \quad \circ\!\!-\!\!\bullet \quad 2\pi \mathrm{i} k_p \hat{g}(\boldsymbol{k}). \tag{2.61}$$

The derivation theorem results directly from the definition of the inverse Fourier transform in Eq. (2.33) by interchanging the partial derivative with the Fourier integral.

The inverse Fourier transform of $2\pi \mathrm{i} k_1$, that is, the corresponding convolution mask, is no longer an ordinary function ($2\pi \mathrm{i} k_1$ is not absolutely integrable) but the derivative of the δ distribution:

$$2\pi \mathrm{i} k \quad \circ\!\!-\!\!\bullet \quad \delta'(x) = \frac{\mathrm{d}\delta(x)}{\mathrm{d}x} = \lim_{a \to 0} \frac{\mathrm{d}}{\mathrm{d}x} \left(\frac{\exp(-\pi x^2/a^2)}{a} \right). \tag{2.62}$$

Of course, the derivation of the δ distribution exists—as all properties of distributions—only in the sense as a limit of a sequence of functions as shown in the preceding equation.

With the knowledge of derivative and shift operators being convolution operators, we can use the properties summarized in Eq. (2.59) to draw some important conclusions. As any convolution operator commutes with the shift operator, convolution is a shift-invariant operation. Furthermore, we can first differentiate a signal and then perform a convolution operation or vice versa and obtain the same result. The properties in Eq. (2.59) are essential for an effective computation of convolution operations.

Central-limit theorem. The central-limit theorem is mostly known for its importance in the theory of probability [134]. However, it also plays an important role for signal processing as it is a rigorous statement of the tendency that cascaded convolution tends to approach Gaussian form ($\propto \exp(-ax^2)$). Because the Fourier transform of the Gaussian is also a Gaussian (≻ R6), this means that *both* the Fourier transform (the transfer function) and the mask of a convolution approach Gaussian shape.

Thus the central-limit theorem is central to the unique role of the Gaussian function for signal processing. The sufficient conditions under which the central-limit theorem is valid can be formulated in different ways. We use here the conditions from [134] and express the theorem with respect to convolution.

Theorem 6 (Central-limit theorem) *Given N functions $h_n(x)$ with a zero mean $\int_{-\infty}^{\infty} x h_n(x)\mathrm{d}x$ and the variance $\sigma_n^2 = \int_{-\infty}^{\infty} x^2 h_n(x)\mathrm{d}x$ with $z = x/\sigma$, $\sigma^2 = \sum_{n=1}^{N} \sigma_n^2$ then*

$$h = \lim_{N\to\infty} h_1 * h_2 * \ldots * h_N \propto \exp(-z^2/2) \tag{2.63}$$

provided that

$$\lim_{N\to\infty} \sum_{n=1}^{N} \sigma_n^2 \to \infty \tag{2.64}$$

and there exists a number $\alpha > 2$ and a finite constant c such that

$$\int_{-\infty}^{\infty} x^{\alpha} h_n(x)\mathrm{d}x < c < \infty \quad \forall n. \tag{2.65}$$

The theorem is of great practical importance because — especially if h is smooth — the Gaussian shape is approximated sufficiently accurately for values of N as low as 5.

Smoothness and compactness. The smoother a function is, the more compact is its Fourier transform. This general rule can be formulated more quantitatively if we express the smoothness by the number of derivatives that are continuous and the compactness by the asymptotic behavior for large values of k. Then we can state: If a function $g(x)$ and its first $n-1$ derivatives are continuous, its Fourier transform decreases at least as rapidly as $|k|^{-(n+1)}$ for large k, that is, $\lim_{|k|\to\infty} |k|^n g(k) = 0$.

As simple examples we can take the box and triangle functions (see next section). The box function is discontinuous ($n = 0$), its Fourier transform, the sinc function, decays with $|k|^{-1}$. In contrast, the triangle function is continuous, but its first derivative is discontinuous. Therefore, its Fourier transform, the sinc^2 function, decays steeper with $|k|^{-2}$. In order to include also impulsive functions (δ distributions) in this relation, we note that the derivative of a discontinuous function becomes impulsive. Therefore, we can state: If the nth derivative of a function becomes impulsive, the function's Fourier transform decays with $|k|^{-n}$.

The relation between smoothness and compactness is an extension of reciprocity between the spatial and Fourier domain. What is strongly localized in one domain is widely extended in the other and vice versa.

Uncertainty relation. This general law of reciprocity finds another quantitative expression in the classical *uncertainty relation* or the *bandwidth-duration product*. This theorem relates the mean square width of a function and its Fourier transform. The mean square width $(\Delta x)^2$ is defined as

$$(\Delta x)^2 = \frac{\int_{-\infty}^{\infty} x^2 \,|g(x)|^2}{\int_{-\infty}^{\infty} |g(x)|^2} - \left(\frac{\int_{-\infty}^{\infty} x \,|g(x)|^2}{\int_{-\infty}^{\infty} |g(x)|^2} \right)^2 . \tag{2.66}$$

It is essentially the variance of $|g(x)|^2$, a measure of the width of the distribution of the "energy" of the signal. The uncertainty relation states:

Theorem 7 (Uncertainty relation) *The product of the variance of $|g(x)|^2$, $(\Delta x)^2$, and of the variance of $|\hat{g}(k)|^2$, $(\Delta k)^2$, cannot be smaller than $1/4\pi$:*

$$\Delta x \Delta k \geq 1/(4\pi). \tag{2.67}$$

The relations between compactness and smoothness and the uncertainty relation give some basic guidance for the design of linear filter (convolution) operators.

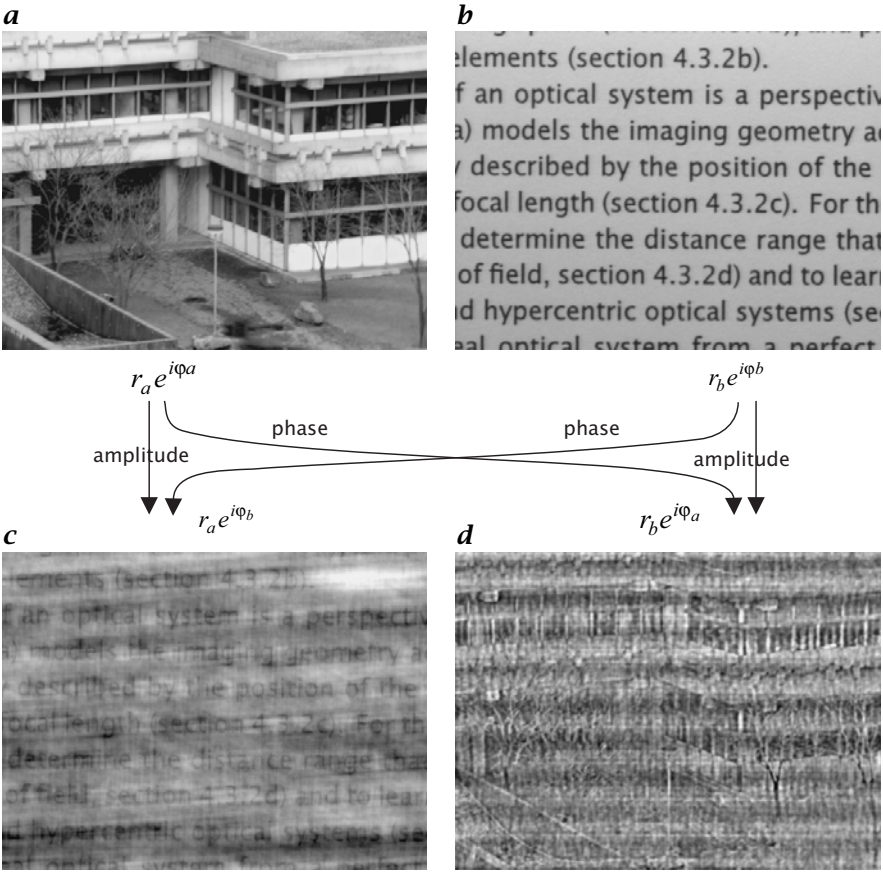

Figure 2.16: *Illustration of the importance of phase and amplitude in Fourier space for the image content:* ***a****,* ***b*** *two original images;* ***c*** *composite image using the phase from image* ***b*** *and the amplitude from image* ***a****;* ***d*** *composite image using the phase from image* ***a*** *and the amplitude from image* ***b****.*

2.3.6 Phase and Amplitude

As outlined above, the DFT can be regarded as a coordinate transformation in a finite-dimensional vector space. Therefore, the image information is completely conserved. The inverse transform results in the original image again.

In Fourier space, we observe the image from another "point of view". Each point in the Fourier domain contains two pieces of information: the *amplitude* and the *phase*, i. e., relative position, of a periodic structure. Given this composition, we ask whether the phase or the amplitude contains the more significant information on the structure in the image, or whether both are of equal importance.

In order to answer this question, we perform a simple experiment. Figure 2.16a, b shows two images. One shows Heidelberg University buildings, the other several lines of printed text. Both images are Fourier transformed and then the phase and amplitude are interchanged as illustrated in Fig. 2.16c, d. The result of this interchange is surprising. It is the phase that determines the content of an image for both images. Both images look patchy but the significant information is preserved.

From this experiment, we can conclude that the phase of the Fourier transform carries essential information about the image structure. The amplitude alone implies only *that* such a periodic structure is contained in the image but not *where*. We can also illustrate this important fact with the *shift theorem* (Theorem 3, p. 52 and ≻ R7). A shift of an object in the space domain leads to a shift of the phase in the wave number domain only. The amplitude is not changed. If we do not know the phase of its Fourier components, we know neither what the object looks like nor where it is located.

It becomes obvious also that the *power spectrum*, i. e., the squared amplitude of the Fourier components (see also Section 3.5.3), contains only very little information, since all the phase information is lost. If a gray value can be associated with the amplitude of a physical process, say a harmonic oscillation, then the power spectrum gives the distribution of the energy in the wave number domain.

2.3.7 Practical Application of the DFT‡

Units. For a practical application of the DFT, it is important to consider the various factors that can be used in the definition of the DFT and to give them a clear meaning. Besides the definition in Eq. (2.29) two others are commonly used:

$$\begin{aligned}
\hat{g}_v &= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} w_N^{-nv}\, g_n \quad \circ\!\!-\!\!\bullet \quad g_n = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} w_N^{nv}\, \hat{g}_v \qquad (a)\\
\hat{g}_v &= \frac{1}{N} \sum_{n=0}^{N-1} w_N^{-nv}\, g_n \quad \circ\!\!-\!\!\bullet \quad g_n = \sum_{n=0}^{N-1} w_N^{nv}\, \hat{g}_v \qquad (b)\\
\hat{g}_v &= \sum_{n=0}^{N-1} w_N^{-nv}\, g_n \quad \circ\!\!-\!\!\bullet \quad g_n = \frac{1}{N} \sum_{n=0}^{N-1} w_N^{nv}\, \hat{g}_v \qquad (c)
\end{aligned} \tag{2.68}$$

Mathematically spoken, the symmetric definition (a) is the most elegant because it uses in both directions the scalar product with the orthonormal base vectors in Eqs. (2.28) and (2.29). In practice, definition (b) is used most often, because $\hat{g}_0$ gives the mean value of the vector in the spatial domain, independent of its length:

$$\hat{g}_0 = \frac{1}{N} \sum_{n=0}^{N-1} w_N^{-n0}\, g_n = \frac{1}{N} \sum_{n=0}^{N-1} g_n. \tag{2.69}$$

Therefore we will use definition (b) almost everywhere in this book.

In practice it is important to know which spatial or temporal intervals have been used to sample the discrete signals. Only then it is possible to compare DFTs correctly that have been sampled with different intervals. The relation can be seen most easily if we approximate the Fourier integral in Eq. (2.18) by a sum and sample the values in the spatial and temporal domain using $x = n\Delta x$, $k = v\Delta k$ und $\Delta x \Delta k = 1/N$:

$$\begin{aligned} \hat{g}(v\Delta k) &= \int_{-\infty}^{\infty} g(x) \exp(-2\pi i v \Delta k x)\, \mathrm{d}x \\ &\approx \sum_{n=0}^{N-1} g_n \exp(-2\pi i n v \Delta x \Delta k)\, \Delta x \\ &= N\Delta x \frac{1}{N} \sum_{n=0}^{N-1} g_n \exp\left(-\frac{2\pi i n v}{N}\right) = N\Delta x \hat{g}_v. \end{aligned} \tag{2.70}$$

These equations state that the Fourier transform g_v computed with the DFT must be multiplied by the factor $N\Delta x = 1/\Delta k$ in order to relate it to a unit interval of the wave number. Without this scaling, the Fourier transform is related to the interval $\Delta k = 1/(N\Delta x)$ and thus differs for signals sampled with different rates.

For 2-D and higher-dimensional signals corresponding relations are valid:

$$\hat{g}(v\Delta k_1, u\Delta k_2) \approx N\Delta x M\Delta y \hat{g}_{uv} = \frac{1}{\Delta k_1 \Delta k_2} \hat{g}_{uv}. \tag{2.71}$$

The same scaling must be applied to the squared signals (*energy*) and not the squared factors from Eq. (2.70). This result follows from the *Rayleigh theorem* for continuous and discrete signals (≻ R4, ≻ R7):

$$\begin{aligned} &\text{Continuous:} \quad \int_{-\infty}^{\infty} |g(x)|^2\, \mathrm{d}x = \int_{-\infty}^{\infty} |\hat{g}(k)|^2\, \mathrm{d}k, \approx \sum_{v=0}^{N-1} |\hat{g}(v\Delta k)|^2\, \Delta k \\ &\text{Discrete:} \quad \frac{1}{N} \sum_{n=0}^{N-1} |g_n|^2 = \sum_{v=0}^{N-1} |\hat{g}_v|^2. \end{aligned} \tag{2.72}$$

The Rayleigh theorem says that the signal energy can either be integrated in the spatial or the Fourier domain. For discrete Signals this means that the average energy is either given by averaging the squared signal in the spatial domain or by summing up the squared magnitude of the signal in the Fourier domain (if we use definition (b) of the DFT in Eq. (2.68)). From the approximation of the integral over the squared magnitude in the Fourier domain by a sum in Eq. (2.72), we can conclude that $|\hat{g}(v\Delta k)|^2 \approx |\hat{g}_v|^2/\Delta k$. The units of the thus scaled squared magnitudes in the Fourier space are $\cdot/m^{-1}$ or $\cdot/Hz$ for *time series*, where $\cdot$ stands for the units of the squared signal.

Dynamic Range. While in most cases it is sufficient to represent an image with 256 quantization levels, i. e., one byte per pixel, the Fourier transform of an image needs a much larger dynamical range. Typically, we observe a strong decrease of the Fourier components with the magnitude of the wave

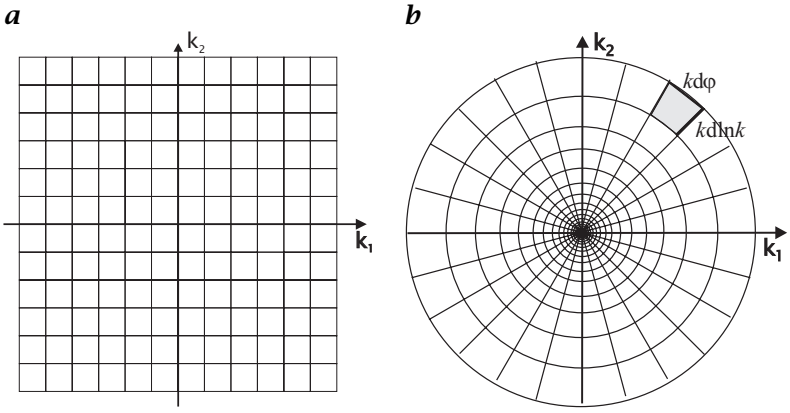

Figure 2.17: *Partition of the Fourier domain into* ***a*** *Cartesian and* ***b*** *logarithmic-polar intervals.*

number (Fig. 2.15). Consequently, at least 16-bit integers or 32-bit floating-point numbers are necessary to represent an image in the Fourier domain without significant rounding errors.

The reason for this behavior is not the insignificance of high wave numbers in images. If we simply omit them, we blur the image. The decrease is caused by the fact that the *relative* resolution is increasing. It is natural to think of relative resolutions, because we are better able to distinguish relative distance differences than absolute ones. We can, for example, easily see the difference of 10 cm in 1 m, but not in 1 km. If we apply this concept to the Fourier domain, it seems to be more natural to represent the images in a so-called *log-polar coordinate system* as illustrated in Fig. 2.17. A discrete grid in this coordinate system separates the space into angular and ln$k$ intervals. Thus the cell area is proportional to k^2. To account for this increase of the area, the Fourier components need to be multiplied by k^2 in this representation:

$$\int_{-\infty}^{\infty} |\hat{g}(\boldsymbol{k})|^2 \mathrm{d}k_1 \mathrm{d}k_2 = \int_{-\infty}^{\infty} k^2 |\hat{g}(\boldsymbol{k})|^2 \mathrm{d}\ln k \mathrm{d}\varphi. \tag{2.73}$$

If we assume that the power spectrum $|\hat{g}(\boldsymbol{k})|^2$ is flat in the natural log-polar coordinate system, it will decrease with k^{-2} in Cartesian coordinates.

For a display of power spectra, it is common to take the logarithm of the gray values in order to compress the high dynamic range. The discussion of log-polar coordinate systems suggests that multiplication by k^2 is a valuable alternative. Likewise, representation in a log-polar coordinate system allows a much better evaluation of the directions of the spatial structures and the smaller scales (Fig. 2.18).

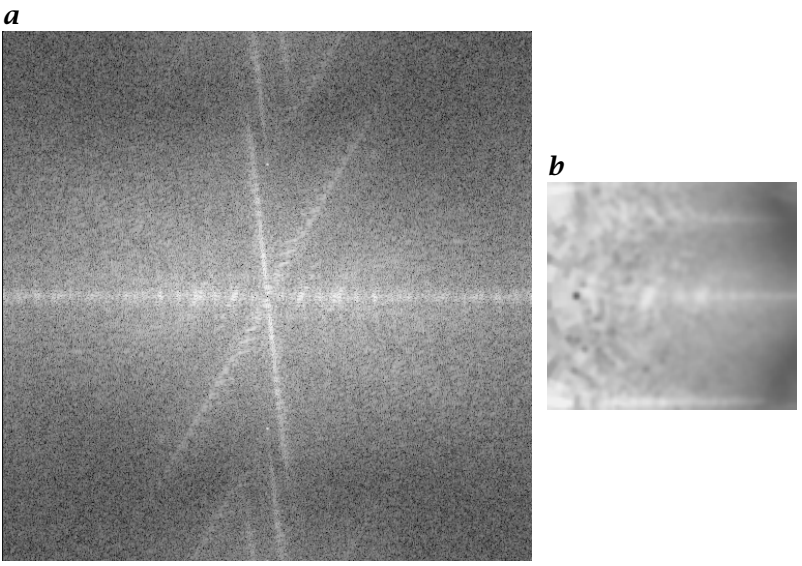

Figure 2.18: *Representation of the Fourier transformed image in Fig. 2.7 in* ***a*** *Cartesian and* ***b*** *logarithmic polar coordinates. Shown is the power spectrum* $|\hat{G}_{uv}|^2$*) multiplied by* k^2*. The gray scale is logarithmic and covers 6 decades (see also Fig. 2.15).*

2.4 Discrete Unitary Transforms‡

2.4.1 General Properties‡

In Sections 2.3.1 and 2.3.2, we learnt that the discrete Fourier transform can be regarded as a linear transformation in a vector space. Thus it is only an example of a large class of transformations, called *unitary transforms*. In this section, we discuss some of their general features which will be of help for a deeper insight into image processing. Furthermore, we give examples of other unitary transforms which have gained importance in digital image processing.

Unitary transforms are defined for vector spaces over the field of complex numbers, for which an *inner product* is defined. Both the FT Eq. (2.22) and DFT Eq. (2.29) basically compute scalar products.

The basic theorem about unitary transform states:

Theorem 8 (Unitary transform) *Let* V *be a finite-dimensional inner product vector space. Let* $\boldsymbol{U}$ *be a one-one linear transformation of* V *onto itself. Then the following are equivalent:*

1. $\boldsymbol{U}$ *is unitary.*
2. $\boldsymbol{U}$ *preserves the inner product, i.e.,* $\langle \boldsymbol{g} | \boldsymbol{h} \rangle = \langle \boldsymbol{U}\boldsymbol{g} | \boldsymbol{U}\boldsymbol{h} \rangle$*,* $\forall \boldsymbol{g}, \boldsymbol{h} \in V$*.*
3. *The inverse of* $\boldsymbol{U}$*,* $\boldsymbol{U}^{-1}$*, is the adjoin of* $\boldsymbol{U}$*:* $\boldsymbol{U}\boldsymbol{U}^{*^T} = \boldsymbol{I}$*.*
4. *The row vectors (and column vectors) of* $\boldsymbol{U}$ *form an orthonormal basis of the vector space* V*.*

In this theorem, the most important properties of a unitary transform are already related to each other: a unitary transform preserves the inner product. This implies that another important property, the *norm*, is also preserved:

$$\|\boldsymbol{g}\|_2 = \langle \boldsymbol{g} \,|\, \boldsymbol{g} \rangle^{1/2} = \langle \boldsymbol{U}\boldsymbol{g} \,|\, \boldsymbol{U}\boldsymbol{g} \rangle^{1/2}. \tag{2.74}$$

It is appropriate to think of the norm as the *length* or *magnitude* of the vector. Rotation in $\mathbb{R}^2$ or $\mathbb{R}^3$ is an example of a transform where the preservation of the length of the vectors is obvious (compare also the discussion of homogeneous coordinates in Section 7.3.3).

The product of two unitary transforms, $\boldsymbol{U}_1\boldsymbol{U}_2$, is unitary. As the identity operator $\boldsymbol{I}$ is unitary, as is the inverse of a unitary operator, the set of all unitary transforms on an inner product space is a *group* under the operation of composition. In practice, this means that we can compose/decompose complex unitary transforms from/into simpler or elementary transforms.

We will illustrate some of the properties of unitary transforms discussed with the discrete Fourier transform. First we consider the one-dimensional DFT Eq. (2.29):

$$\hat{\boldsymbol{g}} = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} g_n \mathrm{w}_M^{-nv}.$$

This equation can be regarded as a multiplication of the $N \times N$ matrix $\boldsymbol{W}_N$

$$(\boldsymbol{W}_N)_{nv} = \mathrm{w}_N^{-nv}$$

with the vector $\boldsymbol{g}$:

$$\hat{\boldsymbol{g}} = \frac{1}{\sqrt{N}} \boldsymbol{W}_N \boldsymbol{g}. \tag{2.75}$$

Explicitly, the DFT for an 8-dimensional vector is given by

$$\begin{bmatrix} \hat{g}_0 \\ \hat{g}_1 \\ \hat{g}_2 \\ \hat{g}_3 \\ \hat{g}_4 \\ \hat{g}_5 \\ \hat{g}_6 \\ \hat{g}_7 \end{bmatrix} = \frac{1}{8} \begin{bmatrix} \mathrm{w}_8^0 & \mathrm{w}_8^0 & \mathrm{w}_8^0 & \mathrm{w}_8^0 & \mathrm{w}_8^0 & \mathrm{w}_8^0 & \mathrm{w}_8^0 & \mathrm{w}_8^0 \\ \mathrm{w}_8^0 & \mathrm{w}_8^7 & \mathrm{w}_8^6 & \mathrm{w}_8^5 & \mathrm{w}_8^4 & \mathrm{w}_8^3 & \mathrm{w}_8^2 & \mathrm{w}_8^1 \\ \mathrm{w}_8^0 & \mathrm{w}_8^6 & \mathrm{w}_8^4 & \mathrm{w}_8^2 & \mathrm{w}_8^0 & \mathrm{w}_8^6 & \mathrm{w}_8^4 & \mathrm{w}_8^2 \\ \mathrm{w}_8^0 & \mathrm{w}_8^5 & \mathrm{w}_8^2 & \mathrm{w}_8^7 & \mathrm{w}_8^4 & \mathrm{w}_8^1 & \mathrm{w}_8^6 & \mathrm{w}_8^3 \\ \mathrm{w}_8^0 & \mathrm{w}_8^4 & \mathrm{w}_8^0 & \mathrm{w}_8^4 & \mathrm{w}_8^0 & \mathrm{w}_8^4 & \mathrm{w}_8^0 & \mathrm{w}_8^4 \\ \mathrm{w}_8^0 & \mathrm{w}_8^3 & \mathrm{w}_8^6 & \mathrm{w}_8^1 & \mathrm{w}_8^4 & \mathrm{w}_8^7 & \mathrm{w}_8^2 & \mathrm{w}_8^5 \\ \mathrm{w}_8^0 & \mathrm{w}_8^2 & \mathrm{w}_8^4 & \mathrm{w}_8^6 & \mathrm{w}_8^0 & \mathrm{w}_8^2 & \mathrm{w}_8^4 & \mathrm{w}_8^6 \\ \mathrm{w}_8^0 & \mathrm{w}_8^1 & \mathrm{w}_8^2 & \mathrm{w}_8^3 & \mathrm{w}_8^4 & \mathrm{w}_8^5 & \mathrm{w}_8^6 & \mathrm{w}_8^7 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \\ g_7 \end{bmatrix}$$

We made use of the periodicity of the kernel of the DFT Eq. (2.43) to limit the exponents of W between 0 and 7. The transformation matrix for the DFT is symmetric ($\boldsymbol{W} = \boldsymbol{W}^T$); $\boldsymbol{W}^{T*}$ is the back transformation.

For the two-dimensional DFT, we can write similar equations if we map the $M \times N$ matrix onto an MN-dimensional vector. There is a simpler way, however, if we make use of the separability of the kernel of the DFT as expressed in Eq. (2.37). Using the $M \times M$ matrix $\boldsymbol{W}_M$ and the $N \times N$ matrix $\boldsymbol{W}_N$ analogously to the one-dimensional case, we can write Eq. (2.75) as

$$\hat{g}_{uv} = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g_{mn} (\boldsymbol{W}_M)_{mu} (\boldsymbol{W}_N)_{nv}, \tag{2.76}$$

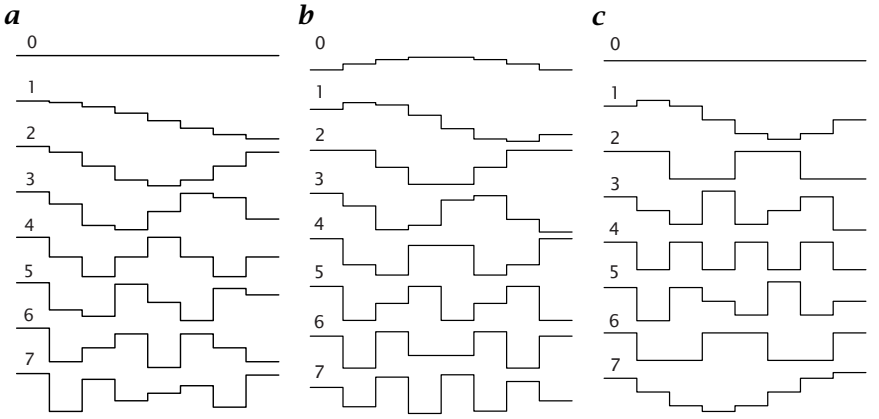

Figure 2.19: *Base functions of one-dimensional unitary transforms for $N = 8$: **a** cosine transform, **b** sine transform, and **c** Hartley transform.*

or, in matrix notation,

$$\underbrace{\hat{\boldsymbol{G}}}_{M\times N} = \frac{1}{\sqrt{MN}} \underbrace{\boldsymbol{W}_M{}^T}_{M\times M} \underbrace{\boldsymbol{G}}_{M\times N} \underbrace{\boldsymbol{W}_N}_{N\times N} = \frac{1}{\sqrt{MN}} \boldsymbol{W}_M \boldsymbol{G} \boldsymbol{W}_N. \tag{2.77}$$

Physicists will be reminded of the theoretical foundations of *quantum mechanics* which are formulated in an inner product vector space of infinite dimension, the *Hilbert space*. In digital image processing, the difficulties associated with infinite-dimensional vector spaces can be avoided.

After discussing the general features, some illustrative examples of unitary transforms will be given. However, none of these transforms is as important as the Fourier transform in signal and image processing.

2.4.2 Cosine, Sine, and Hartley Transforms‡

It is often inconvenient that the discrete Fourier transform maps real-valued to complex-valued images. We can derive a real transformation if we decompose the complex DFT into its real and imaginary parts:

$$(\boldsymbol{W}_N)_{nv} = \cos\left(-\frac{2\pi nv}{N}\right) + \mathrm{i}\sin\left(-\frac{2\pi nv}{N}\right). \tag{2.78}$$

Neither the cosine nor the sine part is useful as a transformation kernel, since these functions do not form a complete basis of the vector space. The cosine and sine functions only span the subspaces of the even and odd functions, respectively.

This problem can be solved by limiting the *cosine transform* and the *sine transform* to the positive half space in the spatial and Fourier domains. Then sym-

metry properties play no role and the two transforms are defined as

$$\begin{aligned} {}^{c}\hat{g}(k) &= \int_0^\infty g(x)\sqrt{2}\cos(2\pi kx)\mathrm{d}x \quad \circ\!\!-\!\!\bullet \quad g(x) = \int_0^\infty {}^{c}\hat{g}(k)\sqrt{2}\cos(2\pi kx)\mathrm{d}k \\ {}^{s}\hat{g}(k) &= \int_0^\infty g(x)\sqrt{2}\sin(2\pi kx)\mathrm{d}x \quad \circ\!\!-\!\!\bullet \quad g(x) = \int_0^\infty {}^{s}\hat{g}(k)\sqrt{2}\sin(2\pi kx)\mathrm{d}k. \end{aligned} \tag{2.79}$$

For the corresponding discrete transforms, base vectors with the missing symmetry can be generated by adding trigonometric functions with half-integer wavelengths. This is equivalent to doubling the base wavelength. Consequently, the kernels for the *cosine* and *sine transforms* in an N-dimensional vector space are

$$c_{nv} = \sqrt{\frac{2}{N}}\cos\left(\frac{\pi nv}{N}\right), \quad s_{nv} = \sqrt{\frac{2}{N+1}}\sin\left(\frac{\pi(n+1)(v+1)}{N+1}\right). \tag{2.80}$$

Figure 2.19a, b shows the base functions of the 1-D cosine and sine functions. From the graphs, it is easy to imagine that all the base functions are orthogonal to each other. Because of the doubling of the periods, both transforms now contain even and odd functions. The base functions with half-integer wavelengths fill in the functions with the originally missing symmetry.

The cosine transform has gained importance for *image data compression* [86]. It is included in the standard compression algorithm proposed by the Joint Photographic Experts Group (*JPEG*).

The *Hartley transform* (*HT*) is a much more elegant solution than the cosine and sine transforms for a transform that avoids complex numbers. By adding the cosine and sine function, we yield an asymmetrical kernel

$$\operatorname{cas} 2\pi kx = \cos(2\pi kx) + \sin(2\pi kx) = \sqrt{2}\cos(2\pi(kx - 1/8)) \tag{2.81}$$

that is suitable for a transform over the whole space domain:

$${}^{h}\hat{g}(k) = \int_{-\infty}^{\infty} g(x)\operatorname{cas}(2\pi kx)\mathrm{d}x \quad \circ\!\!-\!\!\bullet \quad g(x) = \int_{-\infty}^{\infty} {}^{h}\hat{g}(k)\operatorname{cas}(2\pi kx)\mathrm{d}k. \tag{2.82}$$

The corresponding *discrete Hartley transform* (*DHT*) is defined as:

$${}^{h}\hat{g}_v = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} g_n \operatorname{cas}(2\pi nv/N) \quad \circ\!\!-\!\!\bullet \quad g_n = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} {}^{h}\hat{g}_v \operatorname{cas}(2\pi nv/N). \tag{2.83}$$

The base vectors for $N = 8$ are shown in Fig. 2.19c. Despite all elegance of the Hartley transform for real-valued signals, it shows a number of disadvantages in comparison to the Fourier transform. Especially the simple shift theorem 3 of the Fourier trasnform is no longer valid. A shift rather causes base functions with positive and negative wave numbers to be combined with each other:

$$\begin{aligned} g(x - x_0) \quad &\circ\!\!-\!\!\bullet \quad {}^{h}\hat{g}(k)\cos(2\pi kx_0) + {}^{h}\hat{g}(-k)\sin(2\pi kx_0) \\ g_{n-n'} \quad &\circ\!\!-\!\!\bullet \quad {}^{h}\hat{g}_v \cos(2\pi n'v/N) + {}^{h}\hat{g}_{N-v}\sin(2\pi n'v/N) \end{aligned}. \tag{2.84}$$

Similar complications arise with the convolution theorem for the Hartley transform (≻ R8).

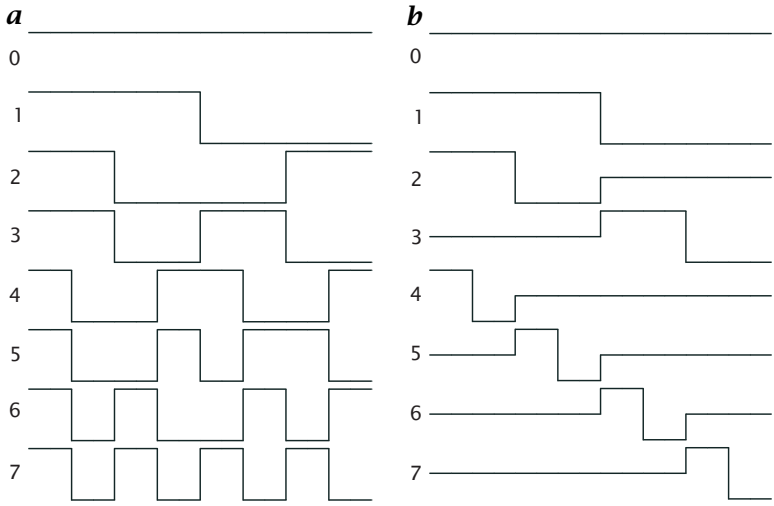

Figure 2.20: *First 8 base functions of one-dimensional unitary transforms for* $N = 16$*:* ***a*** *Hadamard transform and* ***b*** *Haar transform.*

2.4.3 Hadamard Transform‡

The base functions of the *Hadamard transform* are orthogonal binary patterns (Fig. 2.20a). Some of these patterns are regular rectangular waves, others are not. The Hadamard transform is computationally efficient, because its kernel contains only the figures 1 and –1. Thus only additions and subtractions are necessary to compute the transform.

2.4.4 Haar Transform‡

The base vectors of all the transforms considered so far are characterized by the fact that the base functions spread out over the whole vector or image. In this sense we denote these transforms as *global*. All locality is lost. If we have, for example, two independent objects in our image, then they will be simultaneously decomposed into these global patterns and will no longer be recognizable as two individual objects in the transform.

The *Haar transform* is an example of a unitary transform which partly preserves local information, since its base functions are pairs of impulses which are non-zero only at the position of the impulse (Fig. 2.20a). With the Haar transform the position resolution is better for smaller structures. As with the Hadamard transform, the Haar transform is computational efficient. Its kernel only includes the figures -1, 0, and 1.

2.5 Fast Algorithms for Unitary Transforms

2.5.1 Importance of Fast Algorithms

Without an effective algorithm to calculate the discrete Fourier transform, it would not be possible to use the Fourier transform in image processing. Applied directly, Eq. (2.37) is prohibitively expensive. Each point in the transformed image requires N^2 complex multiplications and N^2-1 complex additions (not counting the calculation of the cosine and sine functions in the kernel). In total, we need N^4 complex multiplications and $N^2(N^2-1)$ complex additions. This adds up to about $8N^4$ floating point operations. For a 512×512 image, this results in 5×10^{11} operations. A 200-MHz PentiumPro processor on a PC delivers about 50 MFLOPs (million floating point operations per second) if programmed in a high-level language with an optimizing compiler. A single DFT of a 512×512 image with 5×10^{11} operations would require about 10,000 s or 3 h, much too slow to be of any relevance for practical applications. Thus, the urgent need arises to minimize the number of computations by finding a suitable algorithm. This is an important topic in computer science. To find one we must study the inner structure of the given task, its *computational complexity*, and try to find out how it may be solved with the minimum number of operations.

As an intuitive example, consider the following simple *search* problem. A friend lives in a high-rise building with N floors. We want to find out on which floor his apartment is located. Our questions will only be answered with yes or no. How many questions must we pose to find out where he lives? The simplest and most straightforward approach is to ask "Do you live on floor n?". In the best case, our initial guess is right, but it is more likely to be wrong so that the same question has to be asked with other floor numbers again and again. In the worst case, we must ask exactly $N-1$ questions, in the mean $N/2$ questions. With each question, we can only rule out one out of N possibilities, a quite inefficient approach.

With the question "Do you live in the top half of the building?", however, we can rule out half of the possibilities with just one question. After the answer, we know that he either lives in the top or bottom half, and can continue our questioning in the same manner by splitting up the remaining possibilities into two halves. With this strategy, we need far fewer questions. If the number of floors is a power of two, say 2^l, we need exactly l questions. Thus for N floors, we need $\mathrm{ld}\, N$ questions, where ld denotes the logarithm to the base of two. The strategy applied recursively here for a more efficient solution to the search problem is called *divide and conquer*.

One measure of the computational complexity of a problem with N components is the largest power of N that occurs in the count of op-

erations necessary to solve it. This approximation is useful, since the largest power in N dominates the number of operations necessary for large N. We speak of a zero-order problem $O(N^0)$ if the number of operations does not depend on its size and a linear-order problem $O(N)$ if the number of computations increases linearly with the size. Likewise for solutions. The straightforward solution of the search problem discussed in the previous example is a solution of order N, $O(N)$, the divide-and-conquer strategy is one of $O(\operatorname{ld} N)$.

2.5.2 The 1-D Radix-2 FFT Algorithms

First we consider fast algorithms for the one-dimensional DFT, commonly abbreviated as *FFT* algorithms for *fast Fourier transform*. We assume that the dimension of the vector is a power of two, $N = 2^l$. As the direct solution according to Eq. (2.29) is $O(N^2)$ it seems useful to use the divide-and-conquer strategy. If we can split the transformation into two parts with vectors the size of $N/2$, we reduce the number of operations from N^2 to $2(N/2)^2 = N^2/2$. This procedure can be applied recursively $\operatorname{ld} N$ times, until we obtain a vector of size 1, whose DFT is trivial because nothing at all has to be done. Of course, this procedure only works if the partitioning is possible and the number of additional operations to put the split transforms together is not of a higher order than $O(N)$.

The result of the recursive partitioning is interesting. We do not have to perform a DFT at all. The whole algorithm to compute the DFT has been shifted over to the recursive composition stages. If these compositions are of the order $O(N)$, the computation of the DFT totals to $O(N \operatorname{ld} N)$ since $\operatorname{ld} N$ compositions have to be performed. In comparison to the direct solution of the order $O(N^2)$, this is a tremendous saving in the number of operations. For $N = 2^{10} = 1024$, the number is reduced by a factor of about 100.

We part the vector into two vectors by choosing the even and odd elements separately (Fig. 2.21):

$$\begin{aligned}
\hat{g}_v &= \sum_{n=0}^{N-1} g_n \exp\left(-\tfrac{2\pi i n v}{N}\right) \\
&= \sum_{n=0}^{N/2-1} g_{2n} \exp\left(-\tfrac{2\pi i 2nv}{N}\right) + \sum_{n=0}^{N/2-1} g_{2n+1} \exp\left(-\tfrac{2\pi i (2n+1)v}{N}\right) \\
&= \sum_{n=0}^{N/2-1} g_{2n} \exp\left(-\tfrac{2\pi i n v}{N/2}\right) + \exp\left(-\tfrac{2\pi i v}{N}\right) \sum_{n=0}^{N/2-1} g_{2n+1} \exp\left(-\tfrac{2\pi i n v}{N/2}\right).
\end{aligned} \tag{2.85}$$

Both sums constitute a DFT with $N' = N/2$. The second sum is multiplied by a phase factor which depends only on the wave number v. This

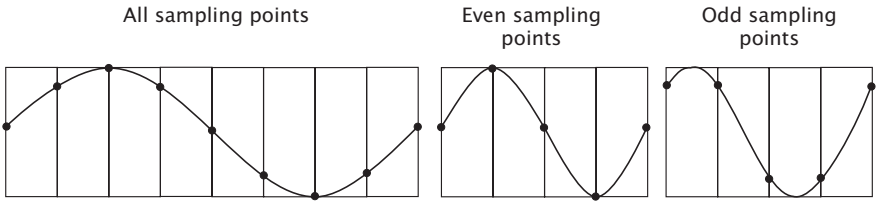

Figure 2.21: *Decomposition of a vector into two vectors containing the even and odd sampling points.*

phase factor results from the shift theorem, since the odd elements are shifted one place to the left.

As an example, we take the base vector with $v = 1$ and $N = 8$ (Fig. 2.21). Taking the odd sampling points, the function shows a phase shift of $\pi/4$. This phase shift is exactly compensated by the phase factor in Eq. (2.85):

$$\exp(-2\pi \mathrm{i} v/N) = \exp(-\pi \mathrm{i}/4).$$

The operations necessary to combine the partial Fourier transforms are just one complex multiplication and addition, i. e., $O(N^1)$. Some more detailed considerations are necessary, however, since the DFT over the half-sized vectors only yields $N/2$ values. In order to see how the composition of the N values works, we study the values for v from 0 to $N/2-1$ and $N/2$ to $N-1$ separately. The partial transformations over the even and odd sampling points are abbreviated by ${}^{e}\hat{g}_v$ and ${}^{o}\hat{g}_v$, respectively. For the first part, we can just take the partitioning as expressed in Eq. (2.85). For the second part, $v' = v + N/2$, only the phase factor changes. The addition of $N/2$ results in a change of sign:

$$\exp\left(-\frac{2\pi \mathrm{i}(v+N/2)}{N}\right) = -\exp\left(-\frac{2\pi \mathrm{i} v}{N}\right) \quad \text{or} \quad \mathrm{w}_N^{-(v+N/2)} = -\mathrm{w}_N^{-v}.$$

Making use of this symmetry we can write

$$\left.\begin{aligned} \hat{g}_v &= {}^{e}\hat{g}_v + \mathrm{w}_N^{-v}\ {}^{o}\hat{g}_v \\ \hat{g}_{v+N/2} &= {}^{e}\hat{g}_v - \mathrm{w}_N^{-v}\ {}^{o}\hat{g}_v. \end{aligned}\right\} \quad 0 \le v < N/2. \tag{2.86}$$

The Fourier transforms for the indices v and $v + N/2$ only differ by the sign of the second term. Thus for the composition of *two* terms we only need *one* complex multiplication. The partitioning is now applied recursively. The two transformations of the $N/2$-dimensional vectors are parted again into two transformations each. We obtain similar expressions as in Eq. (2.85) with the only difference being that the phase factor has doubled to $\exp[-(2\pi \mathrm{i} v)/(N/2)]$. The even and odd parts of the even vector contain the points $\{0, 4, 8, \cdots, N/2-4\}$ and $\{2, 6, 10, \cdots, N/2-2\}$, respectively.

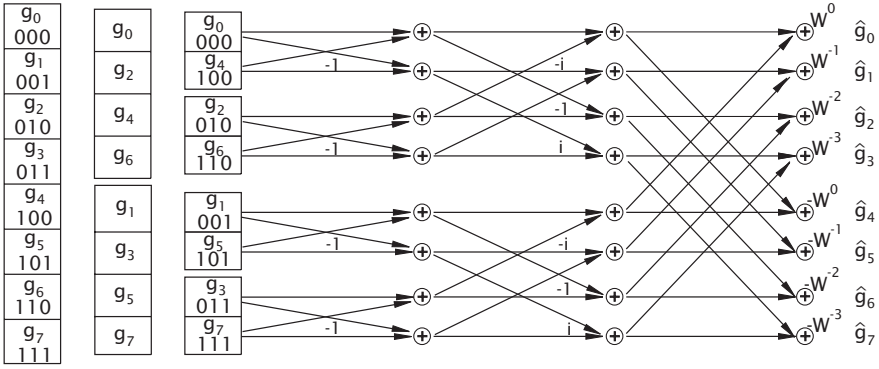

Figure 2.22: *Signal flow diagram of the radix-2 decimation-in-time Fourier transform algorithm for $N = 8$; for further explanation, see text.*

In the last step, we decompose a vector with two elements into two vectors with one element. As the DFT of a single-element vector is an identical operation Eq. (2.29), no further calculations are necessary.

After the decomposition is complete, we can use Eq. (2.86) recursively with appropriate phase factors to compose the original vector step by step in the inverse order. In the first step, we compose vectors with just two elements. Thus we only need the phase factor for $v = 0$ which is equal to one. Consequently, the first composition step has a very simple form:

$$\begin{aligned} \hat{g}_0 &= g_0 + g_1 \\ \hat{g}_{0+N/2} = \hat{g}_1 &= g_0 - g_1. \end{aligned} \tag{2.87}$$

The algorithm we have discussed is called a *decimation-in-time* FFT algorithm, as the signal is decimated in the space domain. All steps of the FFT algorithm are shown in the signal flow diagram in Fig. 2.22 for $N = 8$. The left half of the diagram shows the decimation steps. The first column contains the original vector, the second the result of the first decomposition step into two vectors. The vectors with the even and odd elements are put in the lower and upper halves, respectively. This decomposition is continued until we obtain vectors with one element.

As a result of the decomposition, the elements of the vectors are arranged in a new order. That is all that is performed in the decomposition steps. No computations are required. We can easily understand the new ordering scheme if we represent the indices of the vector with dual numbers. In the first decomposition step we order the elements according to the least significant bit, first the even elements (least significant bit is zero), then the odd elements (least significant bit is one). With each further decomposition step, the bit that governs the sorting is shifted one place to the left. In the end, we obtain a sorting in which the ordering of the bits is completely reversed. The element with the index $1 = 001_2$,

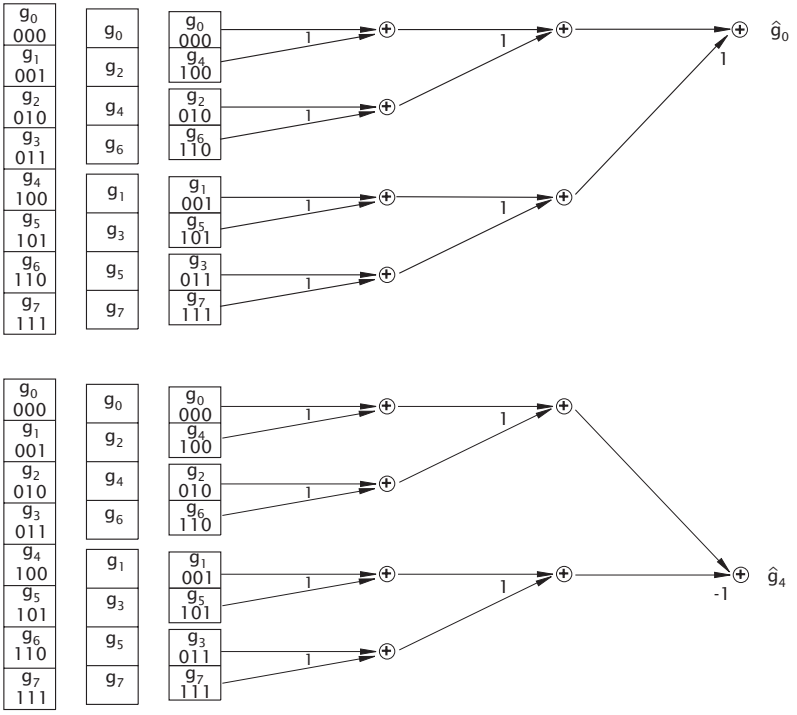

Figure 2.23: *Signal flow path for the calculation of $\hat{g}_0$ and $\hat{g}_4$ with the decimation-in-time FFT algorithm for an 8-dimensional vector.*

for example, will be at the position $4 = 100_2$, and vice versa. Consequently, the chain of decomposition steps can be performed with one operation by interchanging the elements at the normal and bit-reversed positions. This reordering is known as *bit reversal*.

Further steps on the right side of the signal flow diagram show the stepwise composition to vectors of double the size. The composition to the 2-dimensional vectors is given by Eq. (2.87). The operations are pictured with arrows and points having the following meaning: points represent a figure, an element of the vector. These points are called the *nodes* of the signal flow graph. The arrows transfer the figure from one point to another. During the transfer the figure is multiplied by the factor written close to the arrow. If the associated factor is missing, no multiplication takes place. A value of a knot is the sum of the values transferred from the previous level.

The elementary operation of the FFT algorithm involves only two knots. The lower knot is multiplied with a phase factor. The sum and difference of the two values are then transferred to the upper and lower

knot, respectively. Because of the crossover of the signal paths, this operation is denoted as a *butterfly operation*.

We gain further insight into the FFT algorithm if we trace back the calculation of a single element. Figure 2.23 shows the signal paths for $\hat{g}_0$ and $\hat{g}_4$. For each level we go back the number of knots contributing to the calculation doubles. In the last stage all the elements are involved. The signal path for $\hat{g}_0$ and $\hat{g}_4$ are identical but for the last stage, thus nicely demonstrating the efficiency of the FFT algorithm.

All phase factors in the signal path for $\hat{g}_0$ are one. As expected from Eq. (2.29), $\hat{g}_0$ contains the sum of all the elements of the vector $\boldsymbol{g}$,

$$\hat{g}_0 = [(g_0 + g_4) + (g_2 + g_6)] + [(g_1 + g_5) + (g_3 + g_7)],$$

while in the last stage for $\hat{g}_4$ the addition is replaced by a subtraction

$$\hat{g}_4 = [(g_0 + g_4) + (g_2 + g_6)] - [(g_1 + g_5) + (g_3 + g_7)].$$

In Section 2.4, we learnt that the DFT is an example of a unitary transform which is generally performed by multiplying a unitary matrix with the vector. What does the FFT algorithm mean in this context? The signal flow graph in Fig. 2.22 shows that the vector is transformed in several steps. Consequently, the unitary transformation matrix is broken up into several partial transformation matrices that are applied one after the other. If we take the algorithm for $N = 8$ as shown in Fig. 2.22, the unitary matrix is split up into three simpler transformations with sparse unitary transformations:

$$\begin{bmatrix} \hat{g}_0 \\ \hat{g}_1 \\ \hat{g}_2 \\ \hat{g}_3 \\ \hat{g}_4 \\ \hat{g}_5 \\ \hat{g}_6 \\ \hat{g}_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & w^{-1} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & w^{-2} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & w^{-3} \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -w^{-1} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -w^{-2} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -w^{-3} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & \mathrm{i} & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -\mathrm{i} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \mathrm{i} \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\mathrm{i} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \\ g_7 \end{bmatrix}$$

The reader can verify that these transformation matrices reflect all the properties of a single level of the FFT algorithm. The matrix decomposition emphasizes that the FFT algorithm can also be considered as a clever method to decompose the unitary transformation matrix into sparse partial unitary transforms.

2.5.3 Measures for Fast Algorithms

According to the number of arithmetic operations required, there are many other fast Fourier transform algorithms which are still more effective. Most of them are based on polynomial algebra and number theory. An in-depth discussion of these algorithms is given by Blahut [10]. However, the mere number of arithmetic operations is not the only measure for an efficient algorithm. We must also consider a number of other factors.

Access to the data requires additional operations. Consider the simple example of the addition of two vectors. There, besides addition, the following operations are performed: the addresses of the appropriate elements must be calculated; the two elements are read into registers, and the result of these additions is written back to the memory. Depending on the architecture of the hardware used, these five operations constitute a significant overhead which may take much more time than the addition itself. Consequently, an algorithm with a complicated scheme to access the elements of a vector might add a considerable overhead to the arithmetic operations. In effect, a simpler algorithm with more arithmetic operations but less overhead to compute addresses may be faster.

Another factor for rating algorithms is the amount of storage space needed. This not only includes the space for the code but also storage space required for intermediate results or tables for constants. For example, a so-called in-place FFT algorithm, which can perform the Fourier transform on an image without using an intermediate storage area for the image, is very advantageous. Often there is a trade-off between storage space and speed. Many integer FFT algorithms, for example, precalculate the complex phase factors w_N^v and store them in statically allocated tables.

To a large extent the efficiency of algorithms depends on the computer architecture used to implement them. If multiplication is performed either in software or by a microcoded instruction, it is much slower than addition or memory access. In this case, the aim of fast algorithms is to reduce the number of multiplications even at the cost of more additions or a more complex memory access. Such a strategy makes no sense on some modern high-speed architectures where pipelined floating-point addition and multiplication take just one clock cycle. The faster the operations on the processor, the more the memory access becomes the bottleneck. Fast algorithms must now consider effective memory access schemes. It is crucial that as many computations as possible can be performed with one and the same set of data. In this way, these data can be kept in a fast intermediate storage area, known as the *memory cache*, and no direct access to the much slower general memory (RAM) is required.

After this detailed discussion of the algorithm, we can now estimate the number of operations necessary. At each stage of the composition, $N/2$ complex multiplications and N complex additions are carried out. In total we need $N/2 \operatorname{ld} N$ complex multiplications and $N \operatorname{ld} N$ complex additions. A deeper analysis shows that we can save even more multiplications. In the first two composition steps only trivial multiplications by 1 or i occur (compare Fig. 2.22). For further steps the number of trivial multiplications decreases by a factor of two. If our algorithm could avoid all the trivial multiplications, the number of multiplications would be reduced to $(N/2)(\operatorname{ld} N - 3)$.

The FFT algorithm is a classic example of a *fast algorithm*. The computational savings are enormous. For a 512-element vector, only 1536 instead of 262 144 complex multiplications are needed compared to the direct calculation according to Eq. (2.29). The number of multiplications has been reduced by a factor 170.

Using the FFT algorithm, the discrete Fourier transform can no longer be regarded as a computationally expensive operation, since only a few operations are necessary per element of the vector. For a vector with 512 elements, only 3 complex multiplications and 8 complex additions, corresponding to 12 real multiplications and 24 real additions, need to be computed per pixel.

2.5.4 Radix-4 Decimation-in-Time FFT‡

Having worked out one fast algorithm, we still do not know whether the algorithm is optimal or if even more efficient algorithms can be found. Actually, we have applied only one special case of the *divide-and-conquer* strategy. Instead of parting the vector into two pieces, we could have chosen any other partition, say P Q-dimensional vectors, if $N = PQ$. This type of algorithms is called a *Cooley-Tukey algorithm* [10].

Another partition often used is the *radix-4 FFT algorithm*. We can decompose a vector into four components:

$$\begin{aligned}\hat{g}_v &= \sum_{n=0}^{N/4-1} g_{4n} w_N^{-4nv} + w_N^{-v} \sum_{n=0}^{N/4-1} g_{4n+1} w_N^{-4nv} \\ &+ w_N^{-2v} \sum_{n=0}^{N/4-1} g_{4n+2} w_N^{-4nv} + w_N^{-3v} \sum_{n=0}^{N/4-1} g_{4n+3} w_N^{-4nv}.\end{aligned}$$

For simpler equations, we will use similar abbreviations as for the radix-2 algorithm and denote the partial transformations by ${}^0\hat{g}, \cdots, {}^3\hat{g}$. Making use of the symmetry of w_N^v, the transformations into quarters of each of the vectors are given by

$$\begin{aligned}\hat{g}_v &= {}^0\hat{g}_v + w_N^{-v}\,{}^1\hat{g}_v + w_N^{-2v}\,{}^2\hat{g}_v + w_N^{-3v}\,{}^3\hat{g}_v \\ \hat{g}_{v+N/4} &= {}^0\hat{g}_v - \mathrm{i} w_N^{-v}\,{}^1\hat{g}_v - w_N^{-2v}\,{}^2\hat{g}_u + \mathrm{i} w_N^{-3v}\,{}^3\hat{g}_v \\ \hat{g}_{v+N/2} &= {}^0\hat{g}_v - w_N^{-v}\,{}^1\hat{g}_v + w_N^{-2v}\,{}^2\hat{g}_v - w_N^{-3v}\,{}^3\hat{g}_v \\ \hat{g}_{v+3N/4} &= {}^0\hat{g}_v + \mathrm{i} w_N^{-v}\,{}^1\hat{g}_v - w_N^{-2v}\,{}^2\hat{g}_v - \mathrm{i} w_N^{-3v}\,{}^3\hat{g}_v\end{aligned}$$

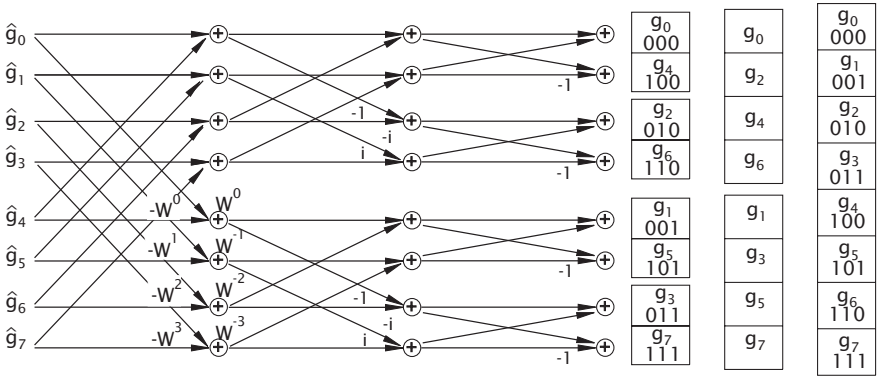

Figure 2.24: *Signal flow diagram of the radix-2 decimation-in-frequency FFT algorithm for $N = 8$.*

or, in matrix notation,

$$\begin{bmatrix} \hat{g}_v \\ \hat{g}_{v+N/4} \\ \hat{g}_{v+N/2} \\ \hat{g}_{v+3N/4} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -\mathrm{i} & -1 & \mathrm{i} \\ 1 & -1 & 1 & -1 \\ 1 & \mathrm{i} & -1 & -\mathrm{i} \end{bmatrix} \begin{bmatrix} {}^0\hat{g}_v \\ \mathrm{w}_N^{-v}\,{}^1\hat{g}_v \\ \mathrm{w}_N^{-2v}\,{}^2\hat{g}_v \\ \mathrm{w}_N^{-3v}\,{}^3\hat{g}_v \end{bmatrix}.$$

To compose 4-tuple elements of the vector, 12 complex additions and 3 complex multiplications are needed. We can reduce the number of additions further by decomposing the matrix into two simpler matrices:

$$\begin{bmatrix} \hat{g}_v \\ \hat{g}_{v+N/4} \\ \hat{g}_{v+N/2} \\ \hat{g}_{v+3N/4} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -\mathrm{i} \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & \mathrm{i} \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} {}^0\hat{g}_v \\ \mathrm{w}_N^{-v}\,{}^1\hat{g}_v \\ \mathrm{w}_N^{-2v}\,{}^2\hat{g}_v \\ \mathrm{w}_N^{-3v}\,{}^3\hat{g}_v \end{bmatrix}.$$

The first matrix multiplication yields intermediate results which can be used for several operations in the second stage. In this way, we save four additions. We can apply this decomposition recursively $\log_4 N$ times. As for the radix-2 algorithm, only trivial multiplications in the first composition step are needed. At all other stages, multiplications occur for 3/4 of the points. In total, $3/4N(\log_4 N - 1) = 3/8N(\mathrm{ld}N - 2)$ complex multiplications and $2N\log_4 N = N\mathrm{ld}N$ complex additions are necessary for the radix-4 algorithm. While the number of additions remains equal, 25 % fewer multiplications are required than for the radix-2 algorithm.

2.5.5 Radix-2 Decimation-in-Frequency FFT‡

The *decimation-in-frequency FFT* is another example of a Cooley-Tukey algorithm. This time, we break the N-dimensional input vector into $N/2$ first and $N/2$ second components. This partition breaks the output vector into its even and odd components:

$$\begin{aligned} \hat{g}_{2v} &= \sum_{n=0}^{N/2-1} (g_n + g_{n+N/2}) w_{N/2}^{-nv} \\ \hat{g}_{2v+1} &= \sum_{n=0}^{N/2-1} W_N^{-n} (g_n - g_{n+N/2}) w_{N/2}^{-nv}. \end{aligned} \tag{2.88}$$

A recursive application of this partition results in a bit reversal of the elements in the output vector, but not the input vector. As an example, the signal flow graph for $N = 8$ is shown in Fig. 2.24. A comparison with the decimation-in-time flow graph (Fig. 2.22) shows that all steps are performed in reverse order. Even the elementary butterfly operations of the decimation-in-frequency algorithm are the inverse of the butterfly operation in the decimation-in-time algorithm.

2.5.6 Multidimensional FFT Algorithms‡

Generally, there are two possible ways to develop fast algorithms for *multidimensional discrete Fourier transforms*. Firstly, we can decompose the multidimensional DFT into 1-D DFTs and use fast algorithms for them. Secondly, we can generalize the approaches of the 1-D FFT for higher dimensions. In this section, we show examples for both possible ways.

Decomposition into 1-D Transforms. A two-dimensional DFT can be broken up in two one-dimensional DFTs because of the separability of the kernel. In the 2-D case Eq. (2.37), we obtain

$$\hat{g}_{u,v} = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \left[\sum_{n=0}^{N-1} g_{m,n} \exp\left(-\frac{2\pi i n v}{N}\right) \right] \exp\left(-\frac{2\pi i m u}{M}\right). \tag{2.89}$$

The inner summation forms M 1-D DFTs of the rows, the outer N 1-D DFTs of the columns, i. e., the 2-D FFT is computed as M row transformations followed by N column transformations

$$\begin{aligned} &\text{Row transform} && \tilde{g}_{m,v} = \frac{1}{N} \sum_{n=0}^{N-1} g_{m,n} \exp\left(-\frac{2\pi i n v}{N}\right) \\ &\text{Column transform} && \hat{g}_{u,v} = \frac{1}{M} \sum_{m=0}^{M-1} \tilde{g}_{m,v} \exp\left(-\frac{2\pi i m u}{M}\right). \end{aligned}$$

In an analogous way, a W-dimensional DFT can be composed of W one-dimensional DFTs.

Multidimensional Decomposition. A decomposition is also directly possible in multidimensional spaces. We will demonstrate such algorithms with the simple case of a 2-D radix-2 decimation-in-time algorithm.
We decompose an $M \times N$ matrix into four submatrices by taking only every second pixel in every second line (Fig. 2.25). This decomposition yields

$$\begin{bmatrix} \hat{g}_{u,v} \\ \hat{g}_{u,v+N/2} \\ \hat{g}_{u+M/2,v} \\ \hat{g}_{u+M/2,v+N/2} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} {}^{0,0}\hat{g}_{u,v} \\ w_N^{-v}\, {}^{0,1}\hat{g}_{u,v} \\ w_M^{-u}\, {}^{1,0}\hat{g}_{u,v} \\ w_M^{-u} w_N^{-v}\, {}^{1,1}\hat{g}_{u,v} \end{bmatrix}.$$

0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3

Figure 2.25: *Decomposition of an image matrix into four partitions for the 2-D radix-2 FFT algorithm.*

The superscripts in front of $\hat{G}$ denote the corresponding partial transformation. The 2-D radix-2 algorithm is very similar to the 1-D radix-4 algorithm. In a similar manner as for the 1-D radix-4 algorithm (Section 2.5.4), we can reduce the number of additions from 12 to 8 by factorizing the matrix:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}.$$

The 2-D radix-2 algorithm for an $N \times N$ matrix requires $(3/4N^2)\,\mathrm{ld}\,N$ complex multiplications, 25 % fewer than the separation into two 1-D radix-2 FFTs. However, the multidimensional decomposition has the disadvantage that the memory access pattern is more complex than for the 1-D Fourier transform. With the partition into a 1-D transform, the access to memory becomes local, yielding a higher cache hit rate than with the distributed access of the multidimensional decomposition.

2.5.7 Transformation of Real Images‡

So far, we have only discussed the Fourier transform of complex-valued signals. The same algorithms can be used also for real-valued signals. Then they are less efficient, however, because the Fourier transform of a real-valued signal is Hermitian (Section 2.3.5) and thus only half of the Fourier coefficients are independent. This corresponds to the fact that also half of the signal, namely the imaginary part, is zero.

It is obvious that another factor two in computational speed can be gained for the DFT of real data. The easiest way to do so is to compute two real 1-D sequences at once. This concept can easily be applied to the DFT of images, because many 1-D DFTs must be computed. Thus we can put the first row $\boldsymbol{x}$ in the real part and the second row $\boldsymbol{y}$ in the imaginary part and yield the complex vector $\boldsymbol{z} = \boldsymbol{x} + \mathrm{i}\boldsymbol{y}$. From the symmetry properties discussed in Section 2.3.5, we infer that the transforms of the real and imaginary parts map in Fourier space

to the Hermitian and anti-Hermitian parts. Thus the Fourier transforms of the two real M-dimensional vectors are given by

$$\hat{x}_v = 1/2(\hat{z}_v + \hat{z}^*_{N-v}), \quad \mathrm{i}\hat{y}_v = 1/2(\hat{z}_v - \hat{z}^*_{N-v}). \tag{2.90}$$

2.6 Further Readings‡

The classical textbook on the Fourier transform — and still one of the best — is Bracewell [11]. An excellent source for various transforms is the "Handbook on Transforms" by Poularikas [141]. For the basics of linear algebra, especially unitary transforms, the reader is referred to one of the modern textbooks on linear algebra, e. g., Meyer [125], Anton [4], or Lay [106]. It is still worthwhile to read the historical article of Cooley and Tukey [22] about the discovery of the first fast Fourier transform algorithm. The monograph of Blahut [10] covers a variety of fast algorithms for the Fourier transform.

3 Random Variables and Fields

3.1 Introduction

Digital image processing can be regarded as a subarea of *digital signal processing*. As such, all the methods for taking and analyzing measurements and their errors can also be applied to image processing. In particular, any measurement we take from images — e. g., the size or the position of an object or its mean gray value — can only be determined with a certain precision and is only useful if we can also estimate its uncertainty. This basic fact, which is well known to any scientist and engineer, was often neglected in the initial days of image processing. Using empirical and ill-founded techniques made reliable error estimates impossible. Fortunately, knowledge in image processing has advanced considerably. Nowadays, many sound image processing techniques are available that include reliable error estimates.

In this respect, it is necessary to distinguish two important classes of errors. The *statistical error* describes the scatter of the measured value if one and the same measurement is repeated over and over again as illustrated in Fig. 3.1. A suitable measure for the width of the distribution gives the statistical error and its centroid, the mean measured value.

This mean value may, however, be much further off the true value than given by the statistical error margins. Such a deviation is called a *systematic error*. Closely related to the difference between systematic and statistical errors are the terms *precise* and *accurate*. A precise but inaccurate measurement is encountered when the statistical error is low but the systematic error is high (Fig. 3.1a). If the reverse is true, i. e., the statistical error is large and the systematic error is low, the individual measurements scatter widely but their mean value is close to the true value (Fig. 3.1b).

It is easy — at least in principle — to get an estimate of the statistical error by repeating the same measurement many times. But it is much harder to control systematic errors. They are often related to a lack in understanding of the measuring setup and procedure. Unknown or uncontrolled parameters influencing the measuring procedure may easily lead to systematic errors. Typical sources of systematic errors are *calibration errors* or temperature-dependent changes of a parameter in an experimental setup without temperature control.

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2
Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

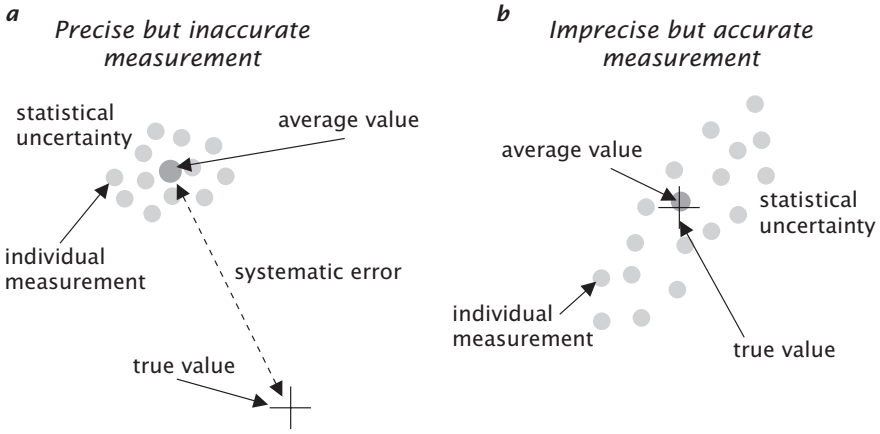

Figure 3.1: *Illustration of* ***a*** *systematic and* ***b*** *statistical error distinguishing precision and accuracy for the measurement of position in 2-D images. The statistical error is given by the distribution of the individual measurements, while the systematic error is the difference between the true value and the average of the measured values.*

In this chapter, we learn how to handle image data as statistical quantities or random variables. We start with the statistical properties of the measured gray value at an individual *sensor element* or *pixel* in Section 3.2. Then we can apply the classical concepts of statistics used to handle point measurements. These techniques are commonly used in most scientific disciplines. The type of statistics used is also known as *first-order statistics* because it considers only the statistics of a single measuring point.

Image processing operations take the measured gray values to compute new quantities. In the simplest case, only the gray value at a single point is taken as an input by so-called *point operations*. In more complex cases, the gray values from many pixels are taken to compute a new point. In any case, we need to know how the statistical properties, especially the precision of the computed quantity depends on the precision of the gray values taken to compute this quantity. In other words, we need to establish how errors are propagating through image processing operations. Therefore, the topic of Section 3.3 is multiple random variables and error propagation.

As a last step, we turn to time series of random variables (*stochastic processes*) and spatial arrays of random variables (*random fields*) in Section 3.5. This allows us to discuss random processes in the Fourier domain.

3.2 Random Variables

3.2.1 Probability Density Functions and Histograms

Imagine an experimental setup in which we are imaging a certain object. The measured quantity at a certain point in the image plane (a pixel) is the irradiance. Because of the statistical nature of the observed process, each measurement will give a different value.

This means that the observed signal is not characterized by a single value but rather a *probability density function* (*PDF*) $f(g)$. This function indicates the probability of observing the value g. A measurable quantity which is governed by a random process is denoted as a *random variable*, or short *RV*.

In the following, we discuss continuous and discrete random variables and probability functions together. We need discrete probabilities as only discrete numbers can be handled by a digital computer. Discrete values are obtained after a process called *quantization* which was introduced in Section 2.2.4. Many equations in this section contain a continuous formulation on the left side and their discrete counterparts on the right side. In the continuous case, $f(g)\mathrm{d}g$ gives the non-negative probability to measure a value within the interval g and $g + \mathrm{d}g$. In the discrete case, we can only measure a finite number, Q, of values g_q ($q = 1, 2, \ldots, Q$) with probability f_q. Normally, the value of a pixel is stored in one byte so that we can measure $Q = 256$ different gray values. As the total probability to observe any value at all is 1 by definition, the PDF must meet the requirement

$$\int_{-\infty}^{\infty} f(g)\mathrm{d}g = 1, \qquad \sum_{q=1}^{Q} f_q = 1. \tag{3.1}$$

The integral of the PDF

$$F(g) = \int_{-\infty}^{g} f(g')\mathrm{d}g' \tag{3.2}$$

is known as the *distribution function*. Because the PDF is a non-negative function, the distribution function increases monotonically from 0 to 1.

Generally, the probability distribution is not known a priori. Rather it is estimated from measurements. If the observed process is *homogeneous*, that is, if it does not depend on the position of the pixel in the image, there is a simple way to estimate the PDF using a *histogram*.

A histogram of an image is a list (vector) which contains one element for each quantization level. Each element contains the number of pixels whose gray value corresponds to the index of the element. Histograms can be calculated easily for data of any dimension. First we set the whole histogram vector to zero. Then we scan each pixel of the image, match

its gray value to an index in the list, and increment the corresponding element of the list by one. The actual scanning algorithm depends on how the image is stored.

An estimate of the probability density function can also be obtained for image data with higher resolution, for instance 16-bit images or floating-point images. Then the range of possible values is partitioned into Q equally wide bins. The value associated with the bin is the center of the bin, whereas we take the values in between the bins to decide which bin is to be incremented. If we do not make this distinction, values computed from the histogram, such as mean values, are biased.

3.2.2 Mean, Variance, and Moments

The two basic parameters that describe a RV g are its *mean* (also known as the *expectation value*) and its *variance*. The mean $\mu = Eg$ is defined as

$$\mu = \int_{-\infty}^{\infty} g f(g) \mathrm{d}g, \qquad \mu = \sum_{q=1}^{Q} g_q f_q. \tag{3.3}$$

The mean can also be determined without knowing the probability density function explictly by averaging an infinite number of measurements:

$$\mu = \lim_{P \to \infty} \frac{1}{P} \sum_{p=1}^{P} g_p. \tag{3.4}$$

As we cannot take an infinite number of measurements, the determination of the mean by Eq. (3.4) remains an estimate with a residual uncertainty that depends on the form of the PDF, i. e., the type of the random process and the number of measurements taken.

The *variance* $\sigma^2 = \operatorname{var} g = E\left((g-\mu)^2\right)$ is a measure of the extent to which the measured values deviate from the mean value:

$$\sigma^2 = \int_{-\infty}^{\infty} (g-\mu)^2 f(g) \mathrm{d}g, \qquad \sigma^2 = \sum_{q=1}^{Q} (g_q - \mu)^2 f_q. \tag{3.5}$$

The PDF can be characterized in more detail by quantities similar to the variance, the *central moments* of nth order $\mu_n = E\left((g-\mu)^n\right)$:

$$\mu_n = \int_{-\infty}^{\infty} (g-\mu)^n f(g) \mathrm{d}g, \qquad \mu_n = \sum_{q=1}^{Q} (g_q - \mu)^n f_q. \tag{3.6}$$

The first central moment is by definition zero. The second moment μ_2 is the variance σ^2. The third moment μ_3, the *skewness*, is a measure for the asymmetry of the PDF around the mean value. If the PDF is a function of even symmetry, $p(-(g-\mu)) = p(g-\mu)$, the third and all higher-order odd moments vanish.

3.2.3 Functions of Random Variables

Any image processing operation changes the signal g at the individual pixels. In the simplest case, g at each pixel is transformed into g' by a function p: $g' = p(g)$. Such a function is known in image processing as a *point operator*. Because g is a RV, g' will also be a RV and we need to know its PDF in order to know the statistical properties of the image after processing it.

It is obvious that the PDF $f_{g'}$ of g' has same form as the PDF f_g of g if p is a linear function: $g' = p_0 + p_1 g$:

$$f_{g'}(g') = \frac{f_g(g)}{|p_1|} = \frac{f_g((g' - p_0)/p_1)}{|p_1|}, \tag{3.7}$$

where the inverse linear relation $g = p^{-1}(g') : g = (g' - p_0)/p_1$ is used to express g as a function of g'.

From Eq. (3.7) it is intuitive that in the general case of a nonlinear function $p(g)$, the slope p_1 will be replaced by the derivative $p'(g_p)$ of $p(g_p)$. Further complications arise if the inverse function has more than one branch. A simple and important example is the function $g' = g^2$ with the two inverse functions $g_{1,2} = \pm\sqrt{g'}$. In such a case, the PDF of g' needs to be added from all branches of the inverse function.

Theorem 9 (PDF of the function of a random variable) *If f_g is the PDF of the random variable g and p a differentiable function $g' = p(g)$, then the PDF of the random variable g' is given by*

$$f_{g'}(g') = \sum_{p=1}^{P} \frac{f_g(g_p)}{\left|p'(g_p)\right|}, \tag{3.8}$$

where g_p are the P real roots of $g' = p(g)$.

A monotonic function p has a unique inverse function $p^{-1}(g')$. Therefore Eq. (3.8) reduces to

$$f_{g'}(g') = \frac{f_g(p^{-1}(g'))}{|p'(p^{-1}(g'))|}. \tag{3.9}$$

In image processing, the following problem is encountered with respect to probability distributions. We have a signal g with a certain PDF and want to transform g by a suitable transform into g' in such a way that g' has a specific probability distribution. This is the inverse problem to what we have discussed so far and it has a surprisingly simple solution. The transform

$$g' = F_{g'}^{-1}(F_g(g)) \tag{3.10}$$

converts the $f_g(g)$-distributed random variable g into the $f_{g'}(g')$-distributed random variable g'. The solution is especially simple for a transformation to a *uniform distribution* because then F^{-1} is a constant function and $g' = F_g(g)$).

Now we consider the mean and variance of functions of random variables. By definition according to Eq. (3.3), the mean of g' is

$$Eg' = \mu_{g'} = \int_{-\infty}^{\infty} g' f_{g'}(g') \mathrm{d}g'. \tag{3.11}$$

We can, however, also express the mean directly in terms of the function $p(g)$ and the PDF $f_g(g)$:

$$Eg' = E\left(p(g)\right) = \int_{-\infty}^{\infty} p(g) f_g(g) \mathrm{d}g. \tag{3.12}$$

Intuitively, you may assume that the mean of g' can be computed from the mean of $g : Eg' = p(Eg)$. This is, however, only possible if p is a linear function. If $p(g)$ is approximated by a polynomial

$$p(g) = p(\mu_g) + p'(\mu_g)(g - \mu_g) + p''(\mu_g)(g - \mu_g)^2/2 + \ldots \tag{3.13}$$

then

$$\mu_{g'} \approx p(\mu_g) + p''(\mu_g)\sigma_g^2/2. \tag{3.14}$$

From this equation we see that $\mu_{g'} \approx p(\mu_g)$ is only a good approximation if the both curvature of $p(g)$ and the variance of g are small.

The first-order estimate of the variance of g' is given by

$$\sigma_{g'}^2 \approx \left| p'(\mu_g) \right|^2 \sigma_g^2. \tag{3.15}$$

This expression is only exact for linear functions p.

The following simple relations for means and variances follow directly from the discussion above (a is a constant):

$$E(ag) = aEg, \quad \mathrm{var}(ag) = a^2 \mathrm{var}\, g, \quad \mathrm{var}\, g = E(g^2) - (Eg)^2. \tag{3.16}$$

3.3 Multiple Random Variables

In image processing, we have many random variables and not just one. Image processing operations compute new values from values at many pixels. Thus, it is important to study the statistics of multiple RVs. In this section, we make the first step and discuss how the statistical properties of multiple RVs and functions of multiple RVs can be handled.

3.3.1 Joint Probability Density Functions

First, we need to consider how the random properties of multiple RVs can be described. Generally, the random properties of two RVs, g_1 and g_2, cannot be described by their individual PDFs, $f(g_1)$ and $f(g_2)$. It is rather necessary to define a *joint probability density function* $f(g_1, g_2)$. Only if the two random variables are *independent*, i. e., if the probability that g_1 takes a certain value does not depend on the value of g_2, we can compute the joint PDF from the individual PDFs, known as *marginal PDFs*:

$$f(g_1, g_2) = f_{g_1}(g_1) f_{g_2}(g_2) \quad \Leftrightarrow \quad g_1, g_2 \text{ independent.} \tag{3.17}$$

For P random variables g_p, the random vector $\boldsymbol{g}$, the joint probability density function is $f(g_1, g_2, \ldots, g_P) = f(\boldsymbol{g})$. The P RVs are called independent if the joint PDF can be written as a product of the marginal PDFs

$$f(\boldsymbol{g}) = \prod_{p=1}^{P} f_{g_p}(g_p) \quad \Leftrightarrow \quad g_p \text{ independent.} \tag{3.18}$$

3.3.2 Covariance and Correlation

The covariance measures to which extent the fluctuations of two RVs, g_p and g_q, are related to each other. In extension of the definition of the *variance* in Eq. (3.5), the *covariance* is defined as

$$C_{pq} = E\left((g_p - \mu_p)(g_q - \mu_q)\right) = E(g_p g_q) - E(g_p)E(g_q). \tag{3.19}$$

For P random variables, the covariances form a $P \times P$ symmetric matrix, the *covariance matrix* $\boldsymbol{C} = \operatorname{cov} \boldsymbol{g}$. The diagonal of this matrix contains the variances of the P RVs.

The *correlation coefficient* relates the covariance to the corresponding variances:

$$r_{pq} = \frac{C}{\sigma_p \sigma_q} \quad \text{where} \quad |r| \leq 1. \tag{3.20}$$

Two RVs g_p and g_q are called *uncorrelated* if the covariance C_{pq} is zero. Then according to Eqs. (3.19) and (3.20) the following relations are true for uncorrelated RVs:

$$C_{12} = 0 \Leftrightarrow r_{12} = 0 \Leftrightarrow E(g_1 g_2) = E(g_1)E(g_2) \Leftrightarrow g_1, g_2 \text{ uncorrelated.} \tag{3.21}$$

From the last of these conditions and Eq. (3.17), it is evident that independent RVs are uncorrelated.

At first glance it appears that only the statistical properties of independent RVs are easy to handle. Then we only need to consider the marginal PDFs of the individual variables together with their mean and

variance. Generally, the interrelation of random variations of the variables as expressed by the covariance matrix $\boldsymbol{C}$ must be considered. Because the covariance matrix is symmetric, however, we can always find a coordinate system, i. e., a linear combination of the RVs, in which the covariance matrix is diagonal and thus the RVs are uncorrelated.

3.3.3 Functions of Multiple Random Variables

In extension to the discussion of functions of a single RV in Section 3.2.3, we can express the mean of a function of multiple random variables $g' = p(g_1, g_2, \ldots, g_P)$ directly from the joint PDF:

$$Eg' = \int_{-\infty}^{\infty} p(g_1, g_2, \ldots, g_P) f(g_1, g_2, \ldots, g_P) \mathrm{d}g_1 \mathrm{d}g_2 \ldots \mathrm{d}g_P. \tag{3.22}$$

From this general relation it follows that the mean of any linear function

$$g' = \sum_{p=1}^{P} a_p g_p \tag{3.23}$$

is given as the linear combination of the means of the RVs g_p:

$$E\left(\sum_{p=1}^{P} a_p g_p\right) = \sum_{p=1}^{P} a_p E\left(g_p\right). \tag{3.24}$$

Note that this is a very general result. We did not assume that the RVs are independent, and this is not dependent on the type of the PDF. As a special case Eq. (3.24) includes the simple relations

$$E(g_1 + g_2) = Eg_1 + Eg_2, \quad E(g_1 + a) = Eg_1 + a. \tag{3.25}$$

The variance of functions of multiple RVs cannot be computed that easy even in the linear case. Let $\boldsymbol{g}$ be a vector of P RVs, $\boldsymbol{g}'$ a vector of Q RVs that is a linear combination of the P RVs $\boldsymbol{g}$, $\boldsymbol{M}$ a $Q \times P$ matrix of coefficients, and $\boldsymbol{a}$ a column vector with Q coefficients. Then

$$\boldsymbol{g}' = \boldsymbol{M}\boldsymbol{g} + \boldsymbol{a} \quad \text{with} \quad E(\boldsymbol{g}') = \boldsymbol{M}E(\boldsymbol{g}) + \boldsymbol{a} \tag{3.26}$$

in extension to Eq. (3.24). If $P = Q$, Eq. (3.26) can be interpreted as a coordinate transformation in a P-dimensional vector space. Therefore it is not surprising that the symmetric covariance matrix transforms as a second-order tensor [134]:

$$\mathrm{cov}(\boldsymbol{g}') = \boldsymbol{M}\,\mathrm{cov}(\boldsymbol{g})\boldsymbol{M}^T. \tag{3.27}$$

To illustrate the application of Eq. (3.27), we apply it to several examples. First, we discuss the computation of the variance of the mean $\overline{g}$ of

P RVs with the same mean and variance σ^2. We assume that the RVs are uncorrelated. Then the matrix $\boldsymbol{M}$ and the covariance matrix $\operatorname{cov}\boldsymbol{g}$ are

$$\boldsymbol{M} = \frac{1}{N}[1,1,1,\ldots,1] \quad \text{and} \quad \operatorname{cov}(\boldsymbol{g}) = \begin{bmatrix} \sigma^2 & 0 & \ldots & 0 \\ 0 & \sigma^2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \sigma^2 \end{bmatrix} = \sigma^2 \boldsymbol{I}.$$

Using these expressions in Eq. (3.27) yields

$$\sigma_{\overline{g}}^2 = \frac{1}{N}\sigma^2. \tag{3.28}$$

Thus the variance $\sigma_{\overline{g}}^2$ is proportional to N^{-1} and the *standard deviation* $\sigma_{\overline{g}}$ decreases only with $N^{-1/2}$. This means that we must take four times as many measurements in order to double the precision of the measurement of the mean. This is not the case for correlated RVs. If the RVs are fully correlated ($r_{pq} = 1$, $C_{pq} = \sigma^2$), according to Eq. (3.27), the variance of the mean is equal to the variance of the individual RVs. In this case it is not possible to reduce the variance by averaging.

In a slight variation, we take P uncorrelated RVs with unequal variances σ_p^2 and compute the variance of the sum of the RVs. From Eq. (3.25), we know already that the mean of the sum is equal to the sum of the means (even for correlated RVs). Similar as for the previous example, it can be shown that for uncorrelated RVs the variance of the sum is also the sum of the individual variances:

$$\operatorname{var} \sum_{p=1}^{P} g_p = \sum_{p=1}^{P} \operatorname{var} g_p. \tag{3.29}$$

As a third example we take Q RVs g'_q that are a linear combination of the P uncorrelated RVs g_p with equal variance σ^2:

$$g'_q = \boldsymbol{a}_q^T \boldsymbol{g}. \tag{3.30}$$

Then the vectors $\boldsymbol{a}_q^T$ form the rows of the $Q \times P$ matrix $\boldsymbol{M}$ in Eq. (3.26) and the covariance matrix of $\boldsymbol{g}'$ results according to Eq. (3.27) in

$$\operatorname{cov}(\boldsymbol{g}') = \sigma^2 \boldsymbol{M}\boldsymbol{M}^T = \sigma^2 \begin{bmatrix} \boldsymbol{a}_1^T\boldsymbol{a}_1 & \boldsymbol{a}_1^T\boldsymbol{a}_2 & \ldots & \boldsymbol{a}_1^T\boldsymbol{a}_Q \\ \boldsymbol{a}_1^T\boldsymbol{a}_2 & \boldsymbol{a}_2^T\boldsymbol{a}_2 & \ldots & \boldsymbol{a}_2^T\boldsymbol{a}_Q \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{a}_1^T\boldsymbol{a}_Q & \boldsymbol{a}_2^T\boldsymbol{a}_Q & \ldots & \boldsymbol{a}_Q^T\boldsymbol{a}_Q \end{bmatrix}. \tag{3.31}$$

From this equation, we can learn two things. First, the variance of the RV g'_q is given by $\boldsymbol{a}_q^T\boldsymbol{a}_q$, i. e., the sum of the squares of the coefficients

$$\sigma^2(g'_q) = \sigma^2 \boldsymbol{a}_q^T \boldsymbol{a}_q. \tag{3.32}$$

Second, although the RVs g_p are uncorrelated, two RVs g'_p and g'_q are only uncorrelated if the scalar product of the coefficient vectors, $\boldsymbol{a}_p^T \boldsymbol{a}_q$, is zero, i. e., the coefficient vectors are orthogonal. Thus, only orthogonal transform matrixes $\boldsymbol{M}$ in Eq. (3.26) leave uncorrelated RVs uncorrelated.

The above analysis of the variance for functions of multiple RVs can be extended to nonlinear functions provided that the function is sufficiently linear around the mean value. As in Section 3.2.3, we expand the nonlinear function $p_q(\boldsymbol{g})$ into a Taylor series around the mean value:

$$g'_p = p_q(\boldsymbol{g}) \approx p_q(\boldsymbol{\mu}) + \sum_{p=1}^{P} \frac{\partial p_q}{\partial g_p}(g_p - \mu_p). \tag{3.33}$$

We compare this equation with Eq. (3.26) and find that the $Q \times P$ matrix $\boldsymbol{M}$ has to be replaced by the matrix $\boldsymbol{J}$

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial p_1}{\partial g_1} & \frac{\partial p_1}{\partial g_2} & \cdots & \frac{\partial p_1}{\partial g_P} \\ \frac{\partial p_2}{\partial g_1} & \frac{\partial p_2}{\partial g_2} & \cdots & \frac{\partial p_2}{\partial g_P} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial p_Q}{\partial g_1} & \frac{\partial p_Q}{\partial g_2} & \cdots & \frac{\partial p_Q}{\partial g_P} \end{bmatrix}, \tag{3.34}$$

known as the *Jacobian matrix* of the transform $\boldsymbol{g}' = \boldsymbol{p}(\boldsymbol{g})$. Thus the covariance of $\boldsymbol{g}'$ is given by

$$\operatorname{cov}(\boldsymbol{g}') \approx \boldsymbol{J} \operatorname{cov}(\boldsymbol{g}) \boldsymbol{J}^T. \tag{3.35}$$

Finally, we discuss the PDFs of function of multiple RVs. We restrict the discussion to two simple cases. First, we consider the addition of two RVs. If two RVs g_1 and g_2 are independent, the resulting probability density function of an additive superposition $g = g_1 + g_2$ is given by the *convolution* integral

$$p_g(g) = \int_{-\infty}^{\infty} p_{g_1}(g') p_{g_2}(g - g') \mathrm{d}g'. \tag{3.36}$$

This general property results from the multiplicative nature of the superposition of probabilities. The probability $p_g(g)$ to measure the value g is the product of the probabilities to measure $g_1 = g'$ and $g_2 = g - g'$. The integral in Eq. (3.36) itself is required because we have to consider all combinations of values that lead to a sum g.

Second, the same procedure can be applied to the multiplication of two RVs if the multiplication of two variables is transformed into an addition by applying the logarithm: $\ln g = \ln g_1 + \ln g_2$. The PDFs of the logarithm of an RV can be computed using Eq. (3.9).

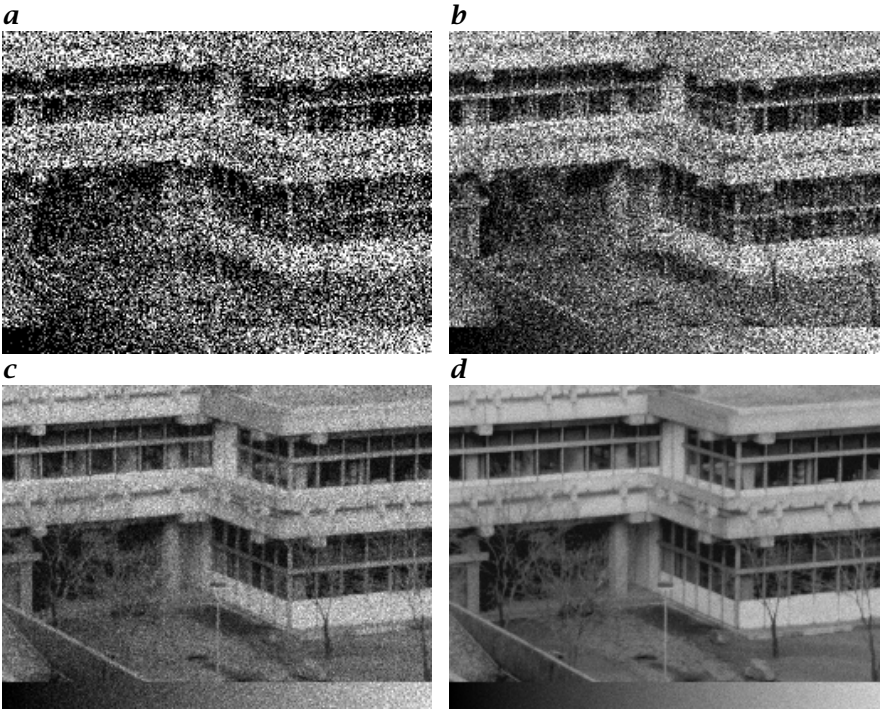

Figure 3.2: *Simulation of low-light images with Poisson noise that have collected maximal* ***a*** *3,* ***b*** *10,* ***c*** *100, and* ***d*** *1000 electrons. Note the linear intensity wedge at the bottom of images* ***c*** *and* ***d*** *.*

3.4 Probability Density Functions

In the previous sections, we derived a number of general properties of random variables without any knowledge about the probability distributions. In this section, we discuss a number of specific probability density functions that are of importance for image processing.

3.4.1 Poisson Distribution

First, we consider image acquisition. An imaging sensor element that is illuminated with a certain irradiance receives within a time interval Δt, the *exposure time*, on average N electrons by absorption of photons. Thus the mean rate of photons per unit time λ is given by

$$\lambda = \frac{N}{\Delta t}. \tag{3.37}$$

Because of the random nature of the stream of photons a different number of photons arrive during each exposure. A random process in which

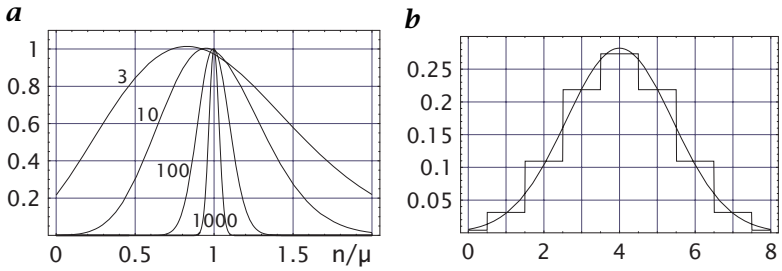

Figure 3.3: ***a*** *Poisson PDFs $P(\mu)$ for mean values μ of 3, 10, 100, and 1000. The x axis is normalized by the mean: the mean value is one; $P(\lambda\Delta t)$ is multiplied by $\sigma\sqrt{2\pi}$;* ***b*** *Discrete binomial PDF $B(8,1/2)$ with a mean of 4 and variance of 2 and the corresponding normal PDF $N(4,2)$.*

we count on average $\lambda\Delta t$ events is known as a *Poisson process* $P(\lambda\Delta t)$. It has the discrete probability density distribution

$$P(\lambda\Delta t): \quad f_n = \exp(-\lambda\Delta t)\frac{(\lambda\Delta t)^n}{n!}, \quad n \geq 0 \tag{3.38}$$

with the mean and variance

$$\mu = \lambda\Delta t \quad \text{and} \quad \sigma^2 = \lambda\Delta t. \tag{3.39}$$

Simulated low-light images with Poisson noise are shown in Fig. 3.2. For low mean values, the Poisson PDF is skewed with a longer tail towards higher values (Fig. 3.3a). But even for a moderate mean (100), the density function is already surprisingly symmetric.

A typical CCD image sensor element (Section 1.7.1, ≻R1) collects in the order of 10000 or more electrons that are generated by absorbed photons. Thus the standard deviation of the number of collected electrons is 100 or 1%. From this figure, we can conclude that even a perfect image sensor element that introduces no additional electronic noise, will show a considerable noise level just by the underlying Poisson process.

The Poisson process has the following important properties:

1. The standard deviation σ is not constant but is equal to the square root of the number of events. Therefore the noise level is signal-dependent.
2. It can be shown that nonoverlapping exposures are statistically independent events [134, Section. 3.4]. This means that we can take images captured with the same sensor at different times as independent RVs.
3. The Poisson process is additive: the sum of two independent Poisson-distributed RVs with the means μ_1 and μ_2 is also Poisson distributed with the mean and variance $\mu_1 + \mu_2$.

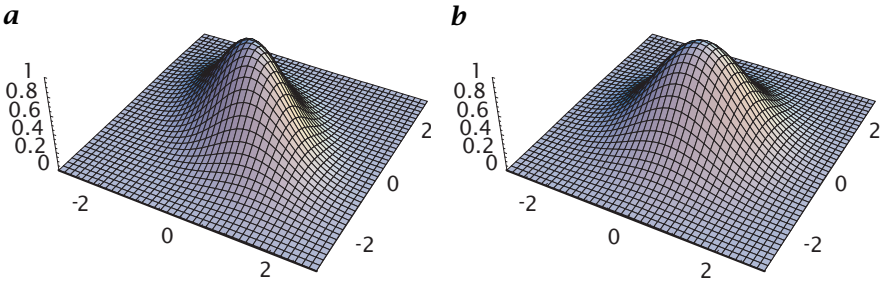

Figure 3.4: *Bivariate normal densities:* ***a*** *correlated RVs with* $\sigma_1^2 = \sigma_2^2 = 1$*, and* $r_{12} = -0.5$*;* ***b*** *isotropic uncorrelated RVs with variances* $\sigma_1^2 = \sigma_1^2 = 1$*.*

3.4.2 Normal and Binomial Distributions

Many processes with continuous RVs can adequately be described by the *normal* or *Gaussian probability density* $N(\mu, \sigma)$ with the mean μ and the variance σ^2:

$$N(\mu,\sigma): \quad f(g) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(g-\mu)^2}{2\sigma^2}\right). \tag{3.40}$$

From Eq. (3.40) we can see that the normal distribution is completely described by the mean and the variance.

For discrete analogue to the normal distribution is the *binomial distribution* $B(Q, p)$

$$B(Q,p): \quad f_q = \frac{Q!}{q!\,(Q-q)!} p^q (1-p)^{Q-q}, \quad 0 \le q < Q. \tag{3.41}$$

The natural number Q denotes the number of possible outcomes and the parameter $p \in]0,1[$ determines together with Q the mean and the variance:

$$\mu = Qp \quad \text{and} \quad \sigma^2 = Qp(1-p). \tag{3.42}$$

Even for moderate Q, the binomial distribution comes very close to the Gaussian distribution as illustrated in Fig. 3.3b.

In extension to Eq. (3.40), the joint normal PDF $N(\boldsymbol{\mu}, \boldsymbol{C})$ for multiple RVs, i.e., the random vector $\boldsymbol{g}$ with the mean $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{C}$ is given by

$$N(\boldsymbol{\mu},\boldsymbol{C}): \; f(\boldsymbol{g}) = \frac{1}{(2\pi)^{P/2}\sqrt{\det \boldsymbol{C}}} \exp\left(-\frac{(\boldsymbol{g}-\boldsymbol{\mu})^T \boldsymbol{C}^{-1} (\boldsymbol{g}-\boldsymbol{\mu})}{2}\right). \tag{3.43}$$

At first glance this expression looks horribly complex. It is not. We must just consider that the symmetric covariance matrix becomes a diagonal matrix by rotation into its principle-axis system. Then the joint

normal density function becomes a separable function

$$f(\boldsymbol{g}') = \prod_{p'=1}^{P} \frac{1}{(2\pi\sigma_{p'}^2)^{1/2}} \exp\left(-\frac{(g_{p'} - \mu_{p'})^2}{2\sigma_{p'}^2}\right) \tag{3.44}$$

with the variances $\sigma_{p'}^2$ along the principle axis (Fig. 3.4a) and the components $g_{p'}$ are independent RVs.

For uncorrelated RVs with equal variance σ^2, the $N(\boldsymbol{\mu}, \boldsymbol{C})$ distribution reduces to the isotropic normal PDF $N(\boldsymbol{\mu}, \sigma)$ (Fig. 3.4b):

$$N(\boldsymbol{\mu}, \sigma): \quad f(\boldsymbol{g}) = \frac{1}{(2\pi\sigma^2)^{P/2}} \exp\left(-\frac{|(\boldsymbol{g} - \boldsymbol{\mu})|^2}{2\sigma^2}\right). \tag{3.45}$$

3.4.3 Central Limit Theorem

The central importance of the normal distribution stems from the *central limit theorem* (Theorem 6, p. 54), which we discussed with respect to cascaded convolution in Section 2.3.5. Here we emphasize its significance for RVs in image processing. The central limit theorem states that under conditions that are almost ever met for image processing applications the PDF of a sum of RVs tends to a normal distribution. As we discussed in Section 3.3, in image processing weighted sums from many values are often computed. Consequently, these combined variables have a normal PDF.

3.4.4 Other Distributions

Despite the significance of the *normal distribution*, other probability density functions also play a certain role for image processing. They occur when RVs are combined by nonlinear functions.

As a first example, we discuss the conversion from *Cartesian* to *polar coordinates*. We take the random vector $\boldsymbol{g} = [g_1, g_2]^T$ with independent $N(0, \sigma)$-distributed components. Then it can be shown [134, Section 6.3] that the magnitude of this vector $r = (g_1^2, g_2^2)^{1/2}$ and the polar angle $\phi = \arctan(g_2/g_1)$ are independent random variables. The magnitude has a *Rayleigh density*

$$R(\sigma): \quad f(r) = \frac{r}{\sigma^2} \exp\left(-\frac{r^2}{2\sigma^2}\right) \quad \text{for} \quad r > 0 \tag{3.46}$$

with the mean and variance

$$\mu_R = \sigma\sqrt{\pi/2} \quad \text{and} \quad \sigma_R^2 = \sigma^2 \frac{4 - \pi}{2}, \tag{3.47}$$

and the angle ϕ has a *uniform density*

$$f(\phi) = \frac{1}{2\pi}. \tag{3.48}$$

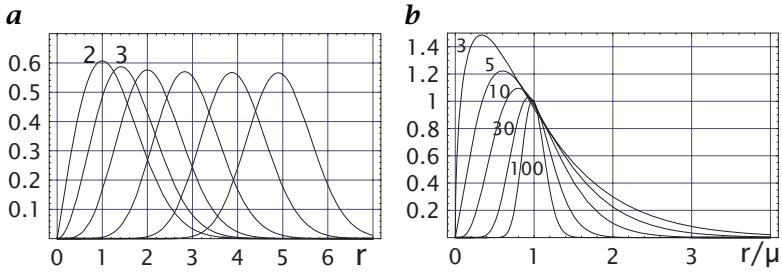

Figure 3.5: ***a*** *Chi density for 2 (Rayleigh density), 3 (Maxwell density), and higher degrees of freedom as indicated;* ***b*** *chi-square density in a normalized plot (mean at one) with degrees of freedom as indicated.*

In generalization of the Rayleigh density, we consider the magnitude of a P dimensional vector. It has a *chi density* with P degrees of freedom

$$\chi(P,\sigma): \quad f(r) = \frac{2r^{P-1}}{2^{P/2}\Gamma(P/2)\sigma^P} \exp\left(-\frac{r^2}{2\sigma^2}\right) \quad \text{for} \quad r > 0 \tag{3.49}$$

with the mean

$$\mu_\chi = \sigma \frac{\sqrt{2}\,\Gamma(P/2+1/2)}{\Gamma(P/2)} \approx \sigma\sqrt{P-1/2} \quad \text{for} \quad P \gg 1 \tag{3.50}$$

and variance

$$\sigma_\chi^2 = \sigma^2 P - \mu_\chi^2 \approx \sigma^2/2 \quad \text{for} \quad P \gg 1. \tag{3.51}$$

The mean of the chi density increases with the square root of P while the variance is almost constant. For large degrees of freedom, the density quickly approaches the normal density $N(\sigma\sqrt{P/2-1/2}, \sigma/\sqrt{2})$ (Fig. 3.5a).

The PDF of the square of the magnitude of the vector has a different PDF because squaring is a nonlinear function (Section 3.2.3). Using Theorem 9 the PDF, known as the *chi-square density* with P degrees of freedom, can be computed as

$$\chi^2(P,\sigma): \quad f(r) = \frac{r^{P/2-1}}{2^{P/2}\Gamma(P/2)\sigma^P} \exp\left(-\frac{r}{2\sigma^2}\right) \quad \text{for} \quad r > 0 \tag{3.52}$$

with the mean and variance

$$\mu_{\chi^2} = \sigma^2 P \quad \text{and} \quad \sigma_{\chi^2}^2 = 2\sigma^4 P \tag{3.53}$$

The sum of squares of RVs is of special importance to obtain the error in the estimation of the *sample variance*

$$s^2 = \frac{1}{P-1}\sum_1^P (g_p - \overline{g})^2 \quad \text{with} \quad \overline{g} = \frac{1}{P}\sum_1^P g_p. \tag{3.54}$$

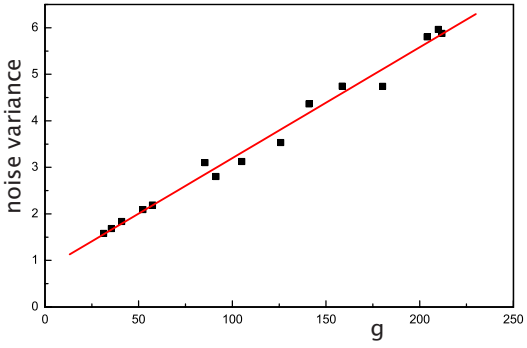

Figure 3.6: *Measured noise variance σ^2 as a function of the gray value g (image courtesy of H. Gröning).*

Papoulis [134, Section 8.2] shows that the normalized sample variance

$$\frac{(P-1)s^2}{\sigma^2} = \sum_{1}^{P} \left(\frac{g_p - \overline{g}}{\sigma} \right)^2 \tag{3.55}$$

has a *chi-square density* with $P-1$ degrees of freedom. Thus the mean of the sample variance is σ^2 (unbiased estimate) and the variance is $2\sigma^4/(P-1)$. For low degrees of freedom, the chi-square density shows significant deviations from the normal density (Fig. 3.5b). For more than 30 degrees of freedom the density is in good approximation normally distributed. A reliable estimate of the variance requires many measurements. For $P = 100$, the relative standard deviation of the variance is still about 20% (for the standard deviation of the standard deviation it is half, 10%).

3.4.5 Noise Model for Image Sensors

After the detailed discussion on random variables, we can now conclude with a simple noise model for an image sensor. In Section 3.4.1 we saw that the photo signal for a single pixel is Poisson distributed. Except for very low-level imaging conditions, where only a few electrons are collected per sensor element, the Poisson distribution is well approximated by a normal distribution $N(Q_e, \sqrt{Q_e})$, where Q_e is the number of electrons absorbed during an exposure. Not every incoming photon causes the excitation of an electron. The fraction of electrons excited by the photons irradiating onto the sensor element (Q_p) is known as *quantum efficiency* η:

$$\eta = \frac{Q_e}{Q_p}. \tag{3.56}$$

The electronic circuits add a number of other noise sources. For practical purposes, it is, however, only important to know that these noise sources are normal distributed and independent of the photon noise. Therefore the received signal g and its variance σ_g^2 can be described by only two terms as

$$\begin{aligned} g &= g_0 + g_p &&= g_0 + \alpha Q_e \quad \text{with} \quad g_p = \alpha Q_e, \\ \sigma_g^2 &= \sigma_0^2 + \sigma_p^2 &&= \sigma_0^2 + \alpha g_p \quad \text{with} \quad \sigma_p^2 = \alpha^2 Q_e. \end{aligned} \tag{3.57}$$

The constant α is an amplification factor of the sensor electronics (including digitization) measured in bits/electron. Equation (3.57) predicts a linear increase of the noise variance with the measured gray value. The measurements with several CCD sensors show good agreement with this model (Fig. 3.6). From the increase of the noise variance σ^2 with the signal g, the amplification factor α can be determined.

3.5 Stochastic Processes and Random Fields‡

The statistics developed so far do not consider the spatial and temporal relations between the points of a multidimensional signal. If we want to analyze the content of images statistically, we must consider the whole image as a statistical quantity, known as a *random field* for spatial data and as a *stochastic process* for time series.

In case of an $M \times N$ image, a random field consists of an $M \times N$ matrix whose elements are random variables. This means that a joint probability density function has MN variables. The mean of a random field is then given as a sum over all possible states q:

$$\overline{G_{m,n}} = \sum_{q=1}^{Q^{MN}} f_q(\boldsymbol{G}) \boldsymbol{G}_q. \tag{3.58}$$

If we have Q quantization levels, each pixel can take Q different states. In combination of all $M \times N$ pixels we end up with Q^{MN} states $\boldsymbol{G}_q$. This is a horrifying concept, rendering itself useless because of the combinatory explosion of possible states. Thus we have to find simpler concepts to treat multidimensional signals as random fields. In this section, we will approach this problem in a practical way.

We start by estimating the mean and variance of a random field. We can do that in the same way as for a single value (Eq. (3.54)), by taking the mean $\boldsymbol{G}_p$ of P measurements under the same conditions and computing the average as

$$\overline{\boldsymbol{G}} = \frac{1}{P} \sum_{p=1}^{P} \boldsymbol{G}_p. \tag{3.59}$$

This type of averaging is known as an *ensemble average*. The estimate of the *variance*, the *sample variance*, is given by

$$\boldsymbol{S}_{\boldsymbol{G}}^2 = \frac{1}{P-1} \sum_{p=1}^{P} \left(\boldsymbol{G}_p - \overline{\boldsymbol{G}} \right)^2. \tag{3.60}$$

At this stage, we know already the mean and variance at each pixel in the image. From these values we can make a number of interesting conclusions. We can study the uniformity of both quantities under given conditions such as a constant illumination level.

3.5.1 Correlation and Covariance Functions‡

In a second step, we now relate the gray values at different positions in the images with each other. One measure for the correlation of the gray values is the mean for the product of the gray values at two positions, the *autocorrelation function*

$$R_{gg}(m,n;m',n') = \overline{G_{mn}G_{m'n'}}. \tag{3.61}$$

As in Eqs. (3.59) and (3.60), an ensemble mean is taken.

The autocorrelation function is not of much use if an image contains a deterministic part with additive *zero-mean noise*

$$\boldsymbol{G}' = \boldsymbol{G} + \boldsymbol{N}, \quad \text{with} \quad \overline{\boldsymbol{G}'} = \boldsymbol{G} \quad \text{and} \quad \overline{\boldsymbol{N}'} = \boldsymbol{0}. \tag{3.62}$$

Then it is more useful to subtract the mean so that the properties of the random part in the signal are adequately characterized:

$$C_{gg}(m,n;m',n') = \overline{(G_{mn} - \overline{G_{mn}})(G_{m'n'} - \overline{G_{m'n'}})}. \tag{3.63}$$

This function is called the *autocovariance function*. For zero shift ($m = m'$ and $n = n'$) it gives the variance at the pixel $[m,n]^T$, at all other shifts the *covariance*, which was introduced in Section 3.3.2, Eq. (3.19). New here is that the autocovariance function includes the spatial relations between the different points in the image. If the autocovariance is zero, the random properties of the corresponding points are uncorrelated.

The autocovariance function as defined in Eq. (3.63) is still awkward because it is four-dimensional. Therefore even this statistic is only of use for a restricted number of shifts, e. g., short distances, because we suspect that the random properties of distant points are uncorrelated.

Things become easier if the statistics do not explicitly depend on the position of the points. This is called a *homogeneous random field*. Then the autocovariance function becomes *shift invariant*:

$$\begin{aligned} C_{gg}(m+k,n+l;m'+k,n'+l) &= C_{gg}(m,n;m',n') \\ &= C_{gg}(m-m',n-n';0,0) \\ &= C_{gg}(0,0;m'-m,n'-n). \end{aligned} \tag{3.64}$$

The last two identities are obtained when we set $(k,l) = (-m',-n')$ and $(k,l) = (-m,-n)$. This also means that the variance of the noise $C_{gg}(m,n;m,n)$ no longer depends on the position in the image but is equal at all points.

Because the autocorrelation function depends only on the distance between points, it reduces from a four- to a two-dimensional function. Fortunately, many stochastic processes are homogeneous. Because of the shift invariance, the

autocovariance function for a homogeneous random field can be estimated by spatial averaging:

$$C_{gg}(m,n) = \frac{1}{MN}\sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1}(G_{m'n'} - \overline{G_{m'n'}})(G_{m'+m,n'+n} - \overline{G_{m'+m,n'+n}}). \quad (3.65)$$

Generally, it is not certain that spatial averaging leads to the same mean as the ensemble mean. A random field that meets this criterion is called *ergodic*.

Another difficulty concerns indexing. As soon as $(m,n) \neq (0,0)$, the indices run over the range of the matrix. We then have to consider the periodic extension of the matrix, as discussed in Section 2.3.5. This is known as *cyclic correlation*.

Now we illustrate the meaning of the autocovariance function. We consider an image that contains a deterministic part plus zero-mean homogeneous noise, see Eq. (3.62). Let us further assume that all points are statistically independent. Then the mean is the deterministic part and the autocovariance vanishes except for zero shift, i. e., for a zero pixel distance:

$$\boldsymbol{C}_{gg} = \sigma^{2\,oo}\boldsymbol{P} \quad \text{or} \quad C_{gg}(m,n) = \sigma^2\delta_m\delta_n. \quad (3.66)$$

For zero shift, the autocovariance is equal to the variance of the noise. In this way, we can examine whether the individual image points are statistically uncorrelated. This is of importance because the degree of correlation between the image points determines the statistical properties of image processing operations as discussed in Section 3.3.3.

In a similar manner to correlating one image with itself, we can correlate two different images $\boldsymbol{G}$ and $\boldsymbol{H}$ with each other. These could be either images from different scenes or images of a dynamic scene taken at different times. By analogy to Eq. (3.65), the *cross-correlation function* and *cross-covariance function* are defined as

$$R_{gh}(m,n) = \frac{1}{MN}\sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1} G_{m'n'}H_{m'+m,n'+n} \quad (3.67)$$

$$C_{gh}(m,n) = \frac{1}{MN}\sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1}(G_{m'n'} - \overline{G_{m'n'}})(H_{m+m',n+n'} - \overline{H_{m+m',n+n'}}). \quad (3.68)$$

The cross-correlation operation is very similar to *convolution* (Section 2.3.5, ≻R7). The only difference is the sign of the indices (m',n') in the second term.

3.5.2 Random Fields in Fourier Space‡

In the previous sections we studied random fields in the spatial domain. Given the significance of the *Fourier transform* for image processing (Section 2.3), we now turn to random fields in the Fourier domain. For the sake of simplicity, we restrict the discussion here to the 1-D case. All arguments put forward in this section can, however, be applied analogously in any dimension.

The Fourier transform requires complex numbers. This constitutes no additional complications, because the random properties of the real and imaginary

part can be treated separately. The definitions for the mean remains the same, the definition of the covariance, however, requires a slight change as compared to Eq. (3.19):

$$C_{pq} = E\left((g_p - \mu_p)^*(g_q - \mu_q)\right), \tag{3.69}$$

where $*$ denotes the conjugate complex. This definition ensures that the variance

$$\sigma_p^2 = E\left((g_p - \mu_p)^*(g_p - \mu_p)\right) \tag{3.70}$$

remains a real number.

The 1-D DFT maps a vector $\boldsymbol{g} \in \mathbb{C}^N$ onto a vector $\hat{\boldsymbol{g}} \in \mathbb{C}^N$. The components of $\hat{\boldsymbol{g}}$ are given as scalar products with orthonormal base vectors for the vector space $\mathbb{C}^N$ (compare Eqs. (2.29) and (2.30)):

$$\hat{g}_v = \boldsymbol{b}_v{}^T \boldsymbol{g} \quad \text{with} \quad \boldsymbol{b}_v{}^T \boldsymbol{b}_{v'} = \delta_{v-v'}. \tag{3.71}$$

Thus the complex RVs in Fourier space are nothing else but linear combinations of the RVs in the spatial domain. If we assume that the RVs in the spatial domain are uncorrelated with equal variance (homogeneous random field), we arrive at a far-reaching conclusion. According to Eq. (3.71) the coefficient vectors $\boldsymbol{b}_v$ are orthogonal to each other with a unit square magnitude. Therefore we can conclude from the discussion about functions of multiple RVs in Section 3.3.3, especially Eq. (3.32), that the RVs in the Fourier domain remain uncorrelated and have the same variance as in the spatial domain.

3.5.3 Power Spectrum, Cross-correlation Spectrum, and Coherence‡

In Section 3.5.1 we learnt that random fields in the space domain are characterized by the auto- and the cross-correlation functions. Now we consider random fields in the Fourier space.

Correlation in the space domain corresponds to multiplication in the Fourier space with the complex conjugate functions (≻ R4):

$$\boldsymbol{G} \star \boldsymbol{G} \circ\!\!-\!\!\bullet\, P_{gg}(\boldsymbol{k}) = \hat{g}(\boldsymbol{k})^* \hat{g}(\boldsymbol{k}) \tag{3.72}$$

and

$$\boldsymbol{G} \star \boldsymbol{H} \circ\!\!-\!\!\bullet\, P_{gh}(\boldsymbol{k}) = \hat{g}(\boldsymbol{k})^* \hat{h}(\boldsymbol{k}). \tag{3.73}$$

In these equations, correlation is abbreviated with the $\star$ symbol, similar to convolution for which we use the $*$ symbol. For a simpler notation, the spectra are written as continuous functions. This corresponds to the transition to an infinitely extended random field (Section 2.3.2, Table 2.1).

The Fourier transform of the autocorrelation function is the *power spectrum* P_{gg}. The power spectrum is a real-valued quantity. Its name is related to the fact that it represents the distribution of power of a physical signal in the Fourier domain, i. e., over frequencies and wave numbers, if the signal amplitude squared is related to the energy of a signal. If the power spectrum is averaged over several images, it constitutes a sum of squares of independent random variables. If the RVs have a normal density, the power spectrum has, according to the discussion in Section 3.4.4, a chi-square density.

The autocorrelation function of a field of uncorrelated RVS is zero except at the origin, i. e., a δ-function (Eq. (3.66)). Therefore, its power spectrum is a constant (≻ R7). This type of noise is called *white noise*.

The Fourier transform of the cross-correlation function is called the *cross-correlation spectrum* P_{gh}. In contrast to the power spectrum, it is a complex quantity. The real and imaginary parts are termed the *co*-co- and *quad-spectrum*, respectively.

To understand the meaning of the cross-correlation spectrum, it is useful to define another quantity, the *coherence function* Φ:

$$\Phi^2(\boldsymbol{k}) = \frac{|P_{gh}(\boldsymbol{k})|^2}{P_{gg}(\boldsymbol{k})P_{hh}(\boldsymbol{k})}. \tag{3.74}$$

Basically, the coherence function contains information on the similarity of two images. We illustrate this by assuming that the image $\boldsymbol{H}$ is a shifted copy of the image $\boldsymbol{G}$: $\hat{h}(\boldsymbol{k}) = \hat{g}(\boldsymbol{k})\exp(-\mathrm{i}\boldsymbol{k}\boldsymbol{x}_s)$. In this case, the coherence function is one and the cross-correlation spectrum P_{gh} reduces to

$$P_{gh}(\boldsymbol{k}) = P_{gg}(\boldsymbol{k})\exp(-\mathrm{i}\boldsymbol{k}\boldsymbol{x}_s). \tag{3.75}$$

Because P_{gg} is a real quantity, we can compute the shift $\boldsymbol{x}_s$ between the two images from the phase factor $\exp(-\mathrm{i}\boldsymbol{k}\boldsymbol{x}_s)$.

If there is no fixed phase relationship of a periodic component between the two images, then the coherency decreases. If the phase shift is randomly distributed from image to image in a sequence, the cross-correlation vectors in the complex plane point in random directions and add up to zero. According to Eq. (3.74), then also the coherency is zero.

3.6 Further Readings‡

An introduction to random signals is given by Rice [149]. A detailed account of the theory of probability and random variables can be found in Papoulis [134]. The textbook of Rosenfeld and Kak [157] gives a good introduction to stochastic processes with respect to image processing. Spectral analysis is discussed in Marple Jr. [119].

4 Neighborhood Operations

4.1 Basic Properties and Purpose

4.1.1 Object Recognition and Neighborhood Operations

An analysis of the spatial relations of the gray values in a small neighborhood provides the first clue for the recognition of objects in images. Let us take a scene containing objects with uniform radiance as a simple example. If the gray value does not change in a small neighborhood, the neighborhood lies within an object. If, however, the gray value changes significantly, an edge of an object crosses the neighborhood. In this way, we recognize areas of constant gray values and edges.

Just processing individual pixels in an image by *point operations* does not provide this type of information. In Chapter 10 we show in detail that such operations are only useful as an initial step of image processing to correct inhomogeneous and nonlinear responses of the imaging sensor, to interactively manipulate images for inspection, or to improve the visual appearance.

A new class of operations is necessary that combines the pixels of a small neighborhood in an appropriate manner and yields a result that forms a new image. Operations of this kind belong to the general class of *neighborhood operations*. These are the central tools for *low-level image processing*. This is why we discuss the principally possible classes of neighborhood operations and their properties in this chapter.

The result of any neighborhood operation is still an image. However, its content has been changed. A properly designed neighborhood operation to detect edges, for instance, should show bright values at pixels that belong to an edge of an object while all pixels — independent of their gray value — should show low values. This example illustrates that by the application of a neighborhood operator, information is generally lost. We can no longer infer the original gray values. This is why neighborhood operations are also called *filters*. They extract a certain *feature* of interest from an image. The image resulting from a neighborhood operator is therefore also called a *feature image*.

It is obvious that operations combining neighboring pixels to form a new image can perform quite different image processing tasks:

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2

Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

- Detection of simple local structures such as edges, corners, lines, and areas of constant gray values (Chapters 12 and 13)
- Motion determination (Chapter 14)
- Texture analysis (Chapter 15)
- *Reconstruction* of images taken with indirect imaging techniques such as *tomography* (Chapter 17)
- *Restoration* of images degraded by defocusing, motion blur, or similar errors during image acquisition (Chapter 17)
- Correction of disturbances caused by errors in image acquisition or transmission. Such errors will result in incorrect gray values for a few individual pixels (Chapter 17)

4.1.2 General Definition

A neighborhood operator N takes the values of the neighborhood around a point, performs some operations with them, and writes the result back on the pixel. This operation is repeated for all points of the signal.

Definition 5 (Continuous neighborhood operator) *A continuous neighborhood operator maps a multidimensional continuous signal $g(\boldsymbol{x})$ onto itself by the following operation*

$$g'(\boldsymbol{x}) = N(\{g(\boldsymbol{x}')\}, \forall(\boldsymbol{x} - \boldsymbol{x}') \in \mathbb{M}) \tag{4.1}$$

where $\mathbb{M}$ is a compact area.

The area $\mathbb{M}$ is called *mask*, *window*, *region of support*, or *structure element* of the neighborhood operation. For the computation of $g'(\boldsymbol{x})$, the size and shape of $\mathbb{M}$ determines the neighborhood operation by specifying the input values of g in the area $\mathbb{M}$ that is shifted with its origin to the point $\boldsymbol{x}$. The neighborhood operation N itself is not specified here. It can be of any type. For symmetry reasons the mask is often symmetric and has its origin in the symmetry center.

Definition 6 (Discrete neighborhood operator) *A discrete neighborhood operator maps a $M \times N$ matrix onto itself by the operation*

$$G'_{m,n} = N(G_{m'-m,n'-n}, \forall\,[m', n']^T \in \mathbb{M}), \tag{4.2}$$

where $\mathbb{M}$ is now a discrete set of points.

Expressions equivalent to Def. 6 can easily be written for dimensions other than two. Although Eqs. (4.1) and (4.2) do not specify in any way the type of neighborhood operation that is performed, they still reveal the common structure of all neighborhood operations.

4.1.3 Mask Size and Symmetry

The first characteristic of a neighborhood operation is the size of the neighborhood. The window may be rectangular or of any other form. We must also specify the position of the pixel relative to the window that will receive the result of the operation. With regard to symmetry, the most natural choice is to place the result of the operation at the pixel in the center of an odd-sized mask of the size $(2R + 1) \times (2R + 1)$.

Even-sized masks seem not to be suitable for neighborhood operations because there is no pixel that lies in the center of the mask. If the result of the neighborhood operation is simply written back to pixels that lay between the original pixels in the center of the mask, we can apply them nevertheless. Thus, the resulting image is shifted by half the pixel distance into every direction. Because of this shift, image features computed by even-sized masks should never be combined with original gray values because this would lead to considerable errors. If we apply several masks in parallel and combine the resulting feature images, all masks must be either even-sized or odd-sized into the same direction. Otherwise, the output lattices do not coincide.

4.1.4 Operator Notation

It is useful to introduce an *operator notation* for neighborhood operators. In this way, complex composite neighbor operations are easily comprehensible. All operators will be denoted by calligraphic letters, such as $\mathcal{B}, \mathcal{D}, \mathcal{H}, \mathcal{S}$. The operator $\mathcal{H}$ transforms the image $\boldsymbol{G}$ into the image $\boldsymbol{G}'$: $\boldsymbol{G}' = \mathcal{H}\boldsymbol{G}$. This notation can be used for continuous and discrete signals of any dimension and leads to a compact *representation-independent notation* of signal-processing operations.

Consecutive application is denoted by writing the operators one after the other. The rightmost operator is applied first. Consecutive application of the same operator is expressed by an exponent

$$\underbrace{\mathcal{H}\mathcal{H}\ldots\mathcal{H}}_{\text{p times}} = \mathcal{H}^p. \tag{4.3}$$

If the operator acts on a single image, the operand, which is to the right in the equations, can be omitted. In this way, operator equations can be written without targets. Furthermore, we will use braces in the usual way to control the order of execution. We can write basic properties of operators in an easily comprehensible way, e. g.,

$$\begin{array}{ll} \text{commutativity} & \mathcal{H}_1\mathcal{H}_2 = \mathcal{H}_2\mathcal{H}_1 \\ \text{associativity} & \mathcal{H}_1(\mathcal{H}_2\mathcal{H}_3) = (\mathcal{H}_1\mathcal{H}_2)\mathcal{H}_3 \\ \text{distributivity over addition} & (\mathcal{H}_1 + \mathcal{H}_2)\mathcal{H}_3 = \mathcal{H}_1\mathcal{H}_2 + \mathcal{H}_2\mathcal{H}_3 \end{array} \tag{4.4}$$

Other operations such as addition can also be used in this operator notation. Care must be taken, however, with any nonlinear operation. As soon as a nonlinear operator is involved, the order in which the operators are executed must strictly be given.

A simple example for a nonlinear operator is pointwise multiplication of images, a dyadic point operator. As this operator occurs frequently, it is denoted by a special symbol, a centered dot ($\cdot$). This symbol is required in order to distinguish it from successive application of operators. The operator expression $\mathcal{B}(\mathcal{D}_p \cdot \mathcal{D}_q)$, for instance, means: apply the operator $\mathcal{D}_p$ and $\mathcal{D}_q$ to the same image, multiply the result pointwise, and apply the operator $\mathcal{B}$ on the product image. Without parentheses the expression $\mathcal{B}\mathcal{D}_p \cdot \mathcal{D}_q$ would mean: apply the operator $\mathcal{D}_q$ to the image and apply the operator $\mathcal{D}_p$ and $\mathcal{B}$ to the same image and then multiply the results point by point. The used operator notation thus gives monadic operators precedence to dyadic operators. If required for clarity, a placeholder for an object onto which an operator is acting is used, denoted by the symbol ":". With a placeholder, the aforementioned operator combination is written as $\mathcal{B}(\mathcal{D}_p : \cdot \mathcal{D}_q :)$.

In the remainder of this chapter we will discuss the two most important classes of neighborhood operations, linear shift-invariant filters (Section 4.2) and rank value filters (Section 4.4). An extra section is devoted to a special subclass of linear-shift-invariant filters, known as recursive filters (Section 4.3).

4.2 Linear Shift-Invariant Filters†

4.2.1 Discrete Convolution

First we focus on the question as to how we can combine the gray values of pixels in a small neighborhood.

The elementary combination of the pixels in the window is given by an operation which multiplies each pixel in the range of the filter mask with the corresponding weighting factor of the mask, adds up the products, and writes the sum to the position of the center pixel:

$$\begin{aligned} g'_{mn} &= \sum_{m'=-r}^{r} \sum_{n'=-r}^{r} h_{m'n'} g_{m-m',n-n'} \\ &= \sum_{m''=-r}^{r} \sum_{n''=-r}^{r} h_{-m'',-n''} g_{m+m',n+n'}. \end{aligned} \tag{4.5}$$

In Section 2.3.5, the *discrete convolution* was defined in Eq. (2.57) as:

$$g'_{mn} = \sum_{m'=0}^{M-1} \sum_{n'=0}^{N-1} h_{m'n'} g_{m-m',n-n'} \tag{4.6}$$

Both definitions are equivalent if we consider the periodicity in the space domain given by Eq. (2.44). From Eq. (2.44) we infer that negative indices are equivalent to positive coefficients by the relations

$$g_{-n} = g_{N-n}, \quad g_{-n,-m} = g_{N-n,M-m}. \tag{4.7}$$

The restriction of the sum in Eq. (4.5) reflects the fact that the elements of the matrix $\boldsymbol{H}$ are zero outside the few points of the $(2R+1) \times (2R+1)$ filter mask. Thus the latter representation is much more practical and gives a better comprehension of the filter operation. The following 3×3 filter mask and the $M \times N$ matrix $\boldsymbol{H}$ are, e. g., equivalent

$$\begin{bmatrix} 0 & -1 & -2 \\ 1 & 0_{\bullet} & -1 \\ 2 & 1 & 0 \end{bmatrix} \equiv \begin{bmatrix} 0_{\bullet} & -1 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & 2 \\ 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 \\ -1 & -2 & 0 & \dots & 0 & 0 \end{bmatrix}. \tag{4.8}$$

A W-dimensional filter operation can be written with a simplified vector indexing:

$$g'_{\boldsymbol{n}} = \sum_{\boldsymbol{n}'=-\boldsymbol{R}}^{\boldsymbol{R}} h_{-\boldsymbol{n}'} g_{\boldsymbol{n}+\boldsymbol{n}'} \tag{4.9}$$

with $\boldsymbol{n} = [n_1, n_2, \dots, n_W]$, $\boldsymbol{R} = [R_1, R_2, \dots, R_W]$, where $h_{\boldsymbol{n}}$ is an element of a W-dimensional signal $g_{n_1,n_2,\dots,n_W}$. The notation for the sums in this equation is an abbreviation for

$$\sum_{\boldsymbol{n}'=-\boldsymbol{R}}^{\boldsymbol{R}} = \sum_{n'_1=-R_1}^{R_1} \sum_{n'_2=-R_2}^{R_2} \dots \sum_{n'_W=-R_W}^{R_W}. \tag{4.10}$$

The vectorial indexing introduced here allows writing most of the relations for arbitrary dimensional signals in a simple way.

4.2.2 Symmetries

With regard to symmetry, we can distinguish two important classes of filters: even and odd filters with the condition in one or more directions that

$$h_{-m,n} = \pm h_{mn} \quad \text{or} \quad h_{m,-n} = \pm h_{mn}, \tag{4.11}$$

where the + and − signs stand for *even* and *odd* symmetry. From this definition we can immediately reduce Eq. (4.5) to make the computation

of one-dimensional filters more efficient:

$$\begin{aligned} &\text{even:} \quad g'_{mn} = h_0 g_{m,n} + \sum_{n'=1}^{r} h_{n'}(g_{m,n-n'} + g_{m,n+n'}) \\ &\text{odd:} \quad g'_{mn} = \sum_{n'=1}^{r} h_{n'}(g_{m,n-n'} - g_{m,n+n'}). \end{aligned} \tag{4.12}$$

The sums only run over half of the filter mask, excluding the center pixel, which must be treated separately because it has no symmetric counterpart. It can be omitted for the odd filter since the coefficient at the center pixel is zero according to Eq. (4.11).

In the 2-D case, the equations become more complex because it is now required to consider the symmetry in each direction separately. A 2-D filter with even symmetry in both directions reduces to

$$\begin{aligned} g'_{m,n} &= h_{00} g_{nm} \\ &+ \sum_{n'=1}^{r} h_{0n'}(g_{m,n-n'} + g_{m,n+n'}) \\ &+ \sum_{m'=1}^{r} h_{m'0}(g_{m-m',n} + g_{m+m',n}) \\ &+ \sum_{m'=1}^{r} \sum_{n'=1}^{r} h_{m'n'}(g_{m-m',n-n'} + g_{m-m',n+n'} \\ &\qquad + g_{m+m',n-n'} + g_{m+m',n+n'}). \end{aligned} \tag{4.13}$$

Also, 2-D filters can have different types of symmetries in different directions. For example, they can be odd in horizontal and even in vertical directions. Then

$$\begin{aligned} g'_{m,n} &= \sum_{n'=1}^{r} h_{0n'}(g_{m,n-n'} - g_{m,n+n'}) \\ &+ \sum_{m'=1}^{r} \sum_{n'=1}^{r} h_{m'n'}(g_{m-m',n-n'} - g_{m-m',n+n'} \\ &\qquad + g_{m+m',n-n'} - g_{m+m',n+n'}). \end{aligned} \tag{4.14}$$

The equations for higher dimensions are even more complex [81].

4.2.3 Computation of Convolution

The discrete convolution operation is such an important operation that it is worth studying it in detail to see how it works. First, we might be confused by the negative signs of the indices m' and n' for either the

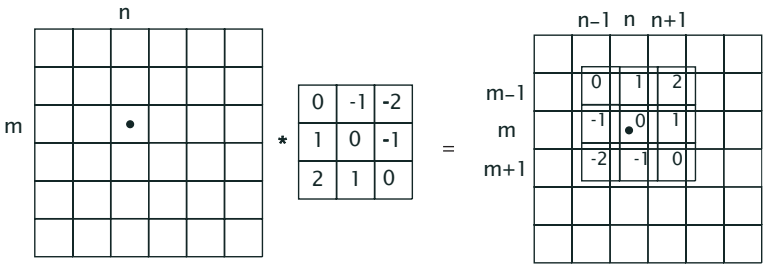

Figure 4.1: *Illustration of the discrete convolution operation with a* 3×3 *filter mask.*

mask or the image in Eq. (4.5). This just means that we mirror either the mask or the image at its symmetry center before we put the mask over the image. We will learn the reason for this mirroring in Section 4.2.5. If we want to calculate the result of the convolution at the point $[m, n]^T$, we center the rotated mask at this point, perform the convolution, and write the result back to position $[m, n]^T$ (Fig. 4.1). This operation is performed for all pixels of the image.

Close to the border of the image, when the filter mask ranges over the edge of the image, we run into difficulties as we are missing some image points. The theoretically correct way to solve this problem according to the periodicity property discussed in Section 2.3.5, especially equation Eq. (2.44), is to take into account that finite image matrices must be thought of as being repeated periodically. Consequently, when we arrive at the left border of the image, we take the missing points from the right edge of the image. We speak of a *cyclic convolution*. Only this type of convolution will reduce to a multiplication in the Fourier space (Section 2.3).

In practice, this approach is seldom chosen because the periodic repetition is artificial, inherently related to the sampling of the image data in Fourier space. Instead we add a border area to the image with half the width of the filter mask. Into this border area we write zeros or we extrapolate in one way or another the gray values from the gray values at the edge of the image. The simplest type of extrapolation is to write the gray values of the edge pixels into the border area. Although this approach gives less visual distortion at the edge of the image than cyclic convolution, we do introduce errors at the edge of the image in a border area with a width of half the size of the filter mask. If we choose any type of extrapolation method, the edge pixels are overweighed. If we set the border area to zero, we introduce horizontal and vertical edges at the image border.

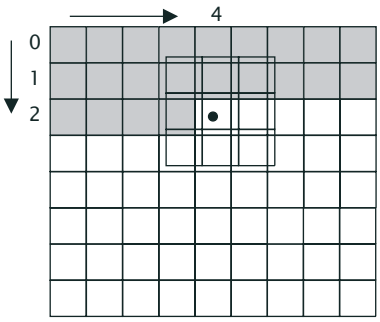

Figure 4.2: *Image convolution by scanning the convolution mask line by line over the image. At the shaded pixels the gray value is already been replaced by the convolution sum. Thus the gray values at the shaded pixels falling within the filter mask need to be stored in an extra buffer.*

In conclusion, no perfect method exists to handle pixels close to edges correctly with neighborhood operations. In one way or another, errors are introduced. The only safe way to avoid errors is to ensure that objects of interest keep a safe distance from the edge of at least half the size of the largest mask used to process the image.

Equation Eq. (4.5) indicates that none of the calculated gray values G'_{mn} will flow into the computation at other neighboring pixels. Thus, if we want to perform the filter operation in-place, we run into a problem. Let us assume that we perform the convolution line by line and from left to right. Then the gray values at all pixel positions above and to the left of the current pixel are already overwritten by the previously computed results (Fig. 4.2).

Consequently, we need to store the gray values at these positions in an appropriate buffer. Efficient algorithms for performing this task are described in Jähne [81] and Jähne et al. [83, Vol. 2, Chap. 5].

The number of elements contained in the mask increases considerably with its size and dimension. A W-dimensional mask with a linear size of R contains R^W elements. The higher the dimension, the faster the number of elements with the size of the mask increases. In higher dimensions, even small neighborhoods include hundreds or thousands of elements.

The challenge for efficient computation schemes is to decrease the number of computations from $O(R^W)$ to a lower order. This means that the number of computations is no longer proportional to R^W but rather to a lower power of R. The ultimate goal is to achieve computation schemes that increase only linearly with the size of the mask ($O(R^1)$) or that do not depend at all on the size of the mask ($O(R^0)$).

4.2.4 Linearity and Shift Invariance

Linear operators are defined by the *principle of superposition*.

Definition 7 (Superposition Principle) *If $\boldsymbol{G}$ and $\boldsymbol{G}'$ are two W-dimensional complex-valued signals, a and b are two complex-valued scalars, and $\mathcal{H}$ is an operator that maps $\boldsymbol{G}$ onto $\boldsymbol{G}'$, then the operator is linear if and only if*

$$\mathcal{H}(a\boldsymbol{G} + b\boldsymbol{G}') = a\mathcal{H}\boldsymbol{G} + b\mathcal{H}\boldsymbol{G}'. \tag{4.15}$$

We can generalize Def. 7 to the superposition of many inputs

$$\mathcal{H}\left(\sum_k a_k \boldsymbol{G}_k\right) = \sum_k a_k \mathcal{H}\boldsymbol{G}_k. \tag{4.16}$$

The superposition states that we can decompose a complex signal into simpler components. We can apply a linear operator to these components and then compose the resulting response from that of the components.

Another important property of an operator is *shift invariance* (also known as *translation invariance* or *homogeneity*). It means that the response of the operator does not explicitly depend on the position in the image. If we shift an image, the output image is the same but for the shift applied. We can formulate this property more elegantly if we define a *shift operator* ${}^{mn}S$ as

$${}^{mn}S g_{m'n'} = g_{m'-m,n'-n}. \tag{4.17}$$

Then we can define a *shift-invariant* operator in the following way:

Definition 8 (Shift invariance) *An operator is shift invariant if and only if it commutes with the shift operator S:*

$$\mathcal{H}\,{}^{mn}S = {}^{mn}S\,\mathcal{H}. \tag{4.18}$$

From the definition of the convolution operation Eqs. (4.5) and (4.9), it is obvious that it is both linear and shift invariant. This class of operators is called *linear shift-invariant operators* (*LSI operators*). In the context of *time series*, the same property is known as *linear time-invariant* (*LTI*). Note that the shift operator ${}^{mn}S$ itself is an LSI operator.

4.2.5 Point Spread Function

The linearity and shift-invariance make it easy to understand the response to a convolution operator. As discussed in Section 2.3.1, we can decompose any discrete image (signal) into its individual points or *basis images*:

$$\boldsymbol{G} = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} G_{mn}\,{}^{mn}\boldsymbol{P}. \tag{4.19}$$

Linearity says that we can apply an operator to each basis image and than add up the resulting images. Shift invariance says, that the response to each of the point images is the same except for a shift. Thus, if we know the response to a point image, we can compute the response to any image.

Consequently, the response to a point image has a special meaning. It is known as the *point spread function* (*PSF*, for time series often denoted as *impulse response*, the response to an impulse). The PSF of a convolution or LSI operator is identical to its mask:

$$p'_{mn} = \sum_{m'=-r}^{r} \sum_{n'=-r}^{r} h_{-m',-n'}\, {}^{00}p_{m+m',n+n'} = h_{m,n} \tag{4.20}$$

and completely describes a convolution operator in the spatial domain.

4.2.6 Transfer Function

In Section 2.3, we discussed that an image can also be represented in the Fourier domain. This representation is of special importance for linear filters since the convolution operation reduces to a multiplication in the Fourier domain according to the *convolution theorem* (Theorem 4, p. 52).

$$\boldsymbol{g} * \boldsymbol{h} \quad \circ\!\!-\!\!\bullet \quad N\hat{\boldsymbol{g}}\hat{\boldsymbol{h}}, \quad \boldsymbol{G} * \boldsymbol{H} \quad \circ\!\!-\!\!\bullet \quad MN\hat{\boldsymbol{G}}\hat{\boldsymbol{H}} \tag{4.21}$$

The Fourier transform of the convolution mask or PSF is known as the *transfer function* of the linear filter. The transfer function has an important practical meaning. For each wave number, it gives the factor by which a periodic structure is multiplied using the filter operation. Note that this factor is a complex number. Thus a periodic structure experiences not only a change in the amplitude but also a phase shift:

$$\begin{aligned} \hat{g}'_{u,v} = \hat{h}_{u,v}\hat{g}_{u,v} &= r_h \exp(\mathrm{i}\varphi_h)\, r_g \exp(\mathrm{i}\varphi_g) \\ &= r_h r_g \exp[\mathrm{i}(\varphi_h + \varphi_g)], \end{aligned} \tag{4.22}$$

where the complex numbers are represented in the second part of the equation with their magnitude and phase as complex exponentials.

The symmetry of the filter masks, as discussed in Section 4.2.2, simplifies the transfer function considerably. We can then combine the corresponding symmetric terms in the Fourier transform of the PSF:

$$\begin{aligned} \hat{h}_v &= \sum_{n'=-R}^{R} h_{n'} \exp\left(-\frac{2\pi \mathrm{i} n v}{N}\right) \quad (\text{with} \quad h_{-n'} = \pm h_{n'}) \\ &= h_0 + \sum_{n'=1}^{r} h_{n'} \left(\exp\left(-\frac{2\pi \mathrm{i} n v}{N}\right) \pm \exp\left(\frac{2\pi \mathrm{i} n v}{N}\right)\right). \end{aligned} \tag{4.23}$$

The definition of the transfer function includes the factors N and MN, respectively. In all future equations for transfer functions Nh_v and $MN\hat{H}$ is replaced by $\hat{h}_v$ and $\hat{H}_v$, respectively.

These equations can be further simplified by replacing the discrete wave number by the scaled continuous wave number

$$\tilde{k} = 2v/N, \quad \text{with} \quad -N/2 \leq v < N/2. \tag{4.24}$$

The scaled wave number $\tilde{k}$ is confined to the interval $[-1, 1[$. A wave number at the edge of this interval corresponds to the maximal wave number that meets the sampling theorem (Section 9.2.3).

Using the Euler equation $\exp(\mathrm{i}x) = \cos x + \mathrm{i}\sin x$, Eq. (4.23) reduces for 1-D even and odd filters to:

$$\begin{aligned} \text{even:} \quad & \hat{h}(\tilde{k}) = h_0 + \sum_{n'=1}^{R} 2h_{n'} \cos(n'\pi\tilde{k}) \\ \text{odd:} \quad & \hat{h}(\tilde{k}) = -\mathrm{i} \sum_{n'=1}^{R} 2h_{n'} \sin(n'\pi\tilde{k}). \end{aligned} \tag{4.25}$$

Correspondingly, a $(2R+1) \times (2R+1)$ mask with even horizontal and vertical symmetry results in the transfer function

$$\begin{aligned} \hat{h}(\tilde{\boldsymbol{k}}) &= h_{00} \\ &+ 2\sum_{n'=1}^{R} h_{0n'} \cos(n'\pi\tilde{k}_1) + 2\sum_{m'=1}^{R} h_{m'0} \cos(m'\pi\tilde{k}_2) \\ &+ 4\sum_{m'=1}^{R}\sum_{n'=1}^{R} h_{m'n'} \cos(n'\pi\tilde{k}_1)\cos(m'\pi\tilde{k}_2). \end{aligned} \tag{4.26}$$

Similar equations are valid for other symmetry combinations.

Equations (4.25) and (4.26) are very useful, because they give a straightforward relationship between the coefficients of a *filter mask* and the *transfer function*. They will be our main tool to study the properties of filters for specific image processing tasks in Chapters 11–15.

4.2.7 Further Properties

In this section, we discuss some further properties of convolution operators that we be useful for image and signal processing.

Property 1 (Commutativity) *LSI operators are commutative:*

$$\mathcal{H}\mathcal{H}' = \mathcal{H}'\mathcal{H}, \tag{4.27}$$

i. e., the order in which we apply convolution operators to an image does not matter. This property is easy to prove in the Fourier domain, because there the operators reduce to a commutative multiplication.

Property 2 (Associativity) *LSI operators are associative:*

$$\mathcal{H}'\mathcal{H}'' = \mathcal{H}. \tag{4.28}$$

Because LSI operations are associative, we can compose a complex operator out of simple operators. Likewise, we can try to decompose a given complex operator into simpler operators. This feature is essential for an effective implementation of convolution operators. As an example, we consider the operator

$$\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}. \tag{4.29}$$

We need 25 multiplications and 24 additions per pixel with this convolution mask. We can easily verify, however, that we can decompose this mask into a horizontal and vertical mask:

$$\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} = [1\ 4\ 6\ 4\ 1] * \begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix}. \tag{4.30}$$

Applying the two convolutions with the smaller masks one after the other, we need only 10 multiplications and 8 additions per pixel. Filter masks which can be decomposed into one-dimensional masks along the axes are called *separable masks*. We will denote one-dimensional operators with an index indicating the axis. We can then write a separable operator $\mathcal{B}$ in a three-dimensional space:

$$\mathcal{B} = \mathcal{B}_z\mathcal{B}_y\mathcal{B}_x. \tag{4.31}$$

In case of one-dimensional masks directed in orthogonal directions, the convolution reduces to an outer product. Separable filters are more efficient the higher the dimension of the space. Let us consider a $9 \times 9 \times 9$ filter mask as an example. A direct implementation would cost 729 multiplications and 728 additions per pixel, while a separable mask of the same size would just need 27 multiplications and 24 additions, a factor of about 30 fewer operations.

Property 3 (Distributivity over Addition) *LSI operators are distributive over addition:*

$$\mathcal{H}' + \mathcal{H}'' = \mathcal{H}. \tag{4.32}$$

Because LSI operators are elements of the same vector space to which they are applied, we can define addition of the operators by the addition of the vector elements. Because of this property we can also integrate operator additions and subtractions into our general operator notation introduced in Section 4.1.4.

4.2.8 Error Propagation with Filtering

Filters are applied to measured data that show noise. Therefore it is important to know how the statistical properties of the filtered data can be inferred from those of the original data. In principal, we solved this question in Section 3.3.3. The *covariance matrix* of the linear combination $\boldsymbol{g}' = \boldsymbol{M}\boldsymbol{g}$ of a random vector $\boldsymbol{g}$ is according to Eq. (3.27) given as

$$\operatorname{cov}(\boldsymbol{g}') = \boldsymbol{M}\operatorname{cov}(\boldsymbol{g})\boldsymbol{M}^T. \tag{4.33}$$

Now we need to apply this result to the special case of a convolution. First, we consider only 1-D signals. We assume that the covaraince matrix of the signal is homogeneous, i. e., depends only on the distance of the points and not the position itself. Then the variance σ^2 for all elements is equal. Furthermore, the values on the side diagonals are also equal and the covariance matrix takes the simple form

$$\operatorname{cov}(\boldsymbol{g}) = \begin{bmatrix} C_0 & C_1 & C_2 & \dots & \dots \\ C_{-1} & C_0 & C_1 & C_2 & \dots \\ C_{-2} & C_{-1} & C_0 & C_1 & \ddots \\ \vdots & C_{-2} & C_{-1} & C_0 & \ddots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{bmatrix}, \tag{4.34}$$

where the index indicates the distance between the points and $C_0 = \sigma^2$. Generally, the covariances decrease with increasing pixel distance. Often, only a limited number of covariances C_n are unequal zero. With statistically uncorrelated pixels, only $C_0 = \sigma^2$ is unequal zero.

Because the linear combinations described by $\boldsymbol{M}$ have the special form of a convolution, the matrix has the same form as the homogeneous covariance matrix. For a filter with three coefficients $\boldsymbol{M}$ reduces to

$$\boldsymbol{M} = \begin{bmatrix} h_0 & h_{-1} & 0 & 0 & 0 \\ h_1 & h_0 & h_{-1} & 0 & 0 \\ 0 & h_1 & h_0 & h_{-1} & 0 \\ 0 & 0 & h_1 & h_0 & \ddots \\ 0 & 0 & 0 & \ddots & \ddots \end{bmatrix}. \tag{4.35}$$

Apart from edge effects, the matrix multiplications in Eq. (4.33) reduce to convolution operations. We introduce the autocovariance vector $\boldsymbol{c} = [\ldots, C_{-1}, C_0, C_1, \ldots]^T$. Then we can write Eq. (4.33) as

$$\boldsymbol{c}' = {}^{-}\boldsymbol{h} * \boldsymbol{c} * \boldsymbol{h} = \boldsymbol{c} * {}^{-}\boldsymbol{h} * \boldsymbol{h} = \boldsymbol{c} \star (\boldsymbol{h} \star \boldsymbol{h}), \tag{4.36}$$

where ${}^{-}\boldsymbol{h}$ is the mirrored convolution mask: ${}^{-}h_n = h_{-n}$. In the last step, we replaced the convolution by a *correlation*. The convolution of $\boldsymbol{c}$ with $\boldsymbol{h} \star \boldsymbol{h}$ can be replced by a correlation, because the autocorrelation function of a real-valued function is a function of even symmetry.

In the case of uncorrelated data, the autocovariance vector is a delta function and the autocovariance vector of the noise of the filtered vector reduces to

$$\boldsymbol{c}' = \sigma^2 (\boldsymbol{h} \star \boldsymbol{h}). \tag{4.37}$$

For a filter with R coefficients, now $2R-1$ values of the autocovariance vector are unequal zero. This means that in the filtered signal pixels with a maximal distance of $R-1$ are now correlated with each other.

Because the covariance vector of a convoluted signal can be described by a correlation, we can also compute the change in the *noise spectrum*, i. e., the *power spectrum* of the noise, caused by a convolution operation. It is just required to Fourier transform Eq. (4.36) under consideration of the correlation theorem (≻ R7). Then we get

$$\boldsymbol{c}' = \boldsymbol{c} \star (\boldsymbol{h} \star \boldsymbol{h}) \quad \circ\!\!-\!\!\bullet \quad \hat{\boldsymbol{c}}'(k) = \hat{\boldsymbol{c}}(k) \left| \hat{\boldsymbol{h}}(k) \right|^2. \tag{4.38}$$

This means that the noise spectrum of a convolved signal is given by the multiplication of the noise spectrum of the input data by the square of the transfer function of the filter. With Eqs. (4.36) and (4.38) we have everything at hand, to compute the changes of the statistical parameters of a signal (variance, autocovariance matrix, and noise spectrum) caused by a filter operation. Going back from Eq. (4.38), we can conclude that Eq. (4.36) is not only valid for 1-D signals but for signals with arbitrary dimensions.

4.2.9 Convolution, Linearity, and Shift Invariance‡

In Section 4.2.4 we saw that a convolution operator is a linear shift invariant operator. But is the reverse also true that *any* linear shift-invariant operator is also a convolution operator? In this section we are going to prove this statement. From our considerations in Section 4.2.5, we are already familiar with the *point spread function* of continuous and discrete operators. Here we introduce the formal definition of the point spread function for an operator $\mathcal{H}$ onto an $M \times N$-dimensional vector space:

$$\boldsymbol{H} = \mathcal{H}\, {}^{00}\boldsymbol{P}. \tag{4.39}$$

Now we can use the linearity Eq. (4.16) and the shift invariance Eq. (4.18) of the operator $\mathcal{H}$ and the definition of the impulse response Eq. (4.39) to calculate the result of the operator on any arbitrary image $\boldsymbol{G}$ in the space domain

$$
\begin{aligned}
(\mathcal{H}\boldsymbol{G})_{mn} &= \left[\mathcal{H}\left[\sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1} g_{m'n'}\, {}^{m'n'}\boldsymbol{P}\right]\right]_{mn} && \text{with Eq. (4.16)} \\
&= \left[\sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1} g_{m'n'}\mathcal{H}\,{}^{m'n'}\boldsymbol{P}\right]_{mn} && \text{linearity} \\
&= \left[\sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1} g_{m'n'}\mathcal{H}\,{}^{m'n'}S^{00}\boldsymbol{P}\right]_{mn} && \text{with Eq. (4.17)} \\
&= \left[\sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1} g_{m'n'}\,{}^{m'n'}S\mathcal{H}\,{}^{00}\boldsymbol{P}\right]_{mn} \\
&= \left[\sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1} g_{m'n'}\,{}^{m'n'}S\boldsymbol{H}\right]_{mn} && \text{with Eq. (4.39)} \\
&= \sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1} g_{m'n'}h_{m-m',n-n'} && \text{with Eq. (4.17)} \\
&= \sum_{m''=0}^{M-1}\sum_{n''=0}^{N-1} g_{m-m'',n-n''}h_{m'',n''} && \begin{matrix} m'' = m - m' \\ n'' = n - n' \end{matrix} .
\end{aligned}
$$

These calculations prove that a linear shift-invariant operator must necessarily be a convolution operation in the space domain. There is no other operator type which is both linear and shift invariant.

4.2.10 Inverse Operators‡

Can we invert a filter operation so that we can get back the original image from a filtered image? This question is significant because degradations such as image blurring by motion or by defocused optics can also be regarded as filter operations (Section 7.6.1). If an inverse operator exists and if we know the point spread function of the degradation, we can reconstruct the original, undisturbed image. The problem of inverting a filter operation is known as *deconvolution* or *inverse filtering*.

By considering the filter operation in the Fourier domain, we immediately recognize that we can only reconstruct those wave numbers for which the transfer function of the filter does not vanish. In practice, the condition for inversion of a filter operation is much more restricted because of the limited quality of the image signals. If a wave number is attenuated below a critical level, which depends on the noise and quantization (Section 9.4), it will not be recoverable. It is obvious that these conditions limit the power of a straightforward inverse filtering considerably. The problem of inverse filtering is considered further in Chapter 17.8.

4.2.11 Eigenfunctions‡

Next we are interested in the question whether special types of images $\boldsymbol{E}$ exist which are preserved by a linear shift-invariant operator, except for multiplication with a scalar. Intuitively, it is clear that these images have a special importance for LSI operators. Mathematically speaking, this means

$$\mathcal{H}\boldsymbol{E} = \lambda\boldsymbol{E}. \tag{4.40}$$

A vector (image) which meets this condition is called an *eigenvector* (*eigenimage*) or *characteristic vector* of the operator, the scaling factor λ an *eigenvalue* or *characteristic value* of the operator.

In order to find out the eigenimages of LSI operators, we discuss the shift operator S. It is quite obvious that for real images only a trivial eigenimage exists, namely a constant image. For complex images, however, a whole set of eigenimages exists. We can find it when we consider the shift property of the *complex exponential*

$$^{uv}w_{mn} = \exp\left(\frac{2\pi imu}{M}\right)\exp\left(\frac{2\pi inv}{N}\right), \tag{4.41}$$

which is given by

$$^{kl}S\,^{uv}\boldsymbol{W} = \exp\left(-\frac{2\pi iku}{M}\right)\exp\left(-\frac{2\pi ilv}{N}\right)\,^{uv}\boldsymbol{W}. \tag{4.42}$$

The latter equation directly states that the complex exponentials $^{uv}\boldsymbol{W}$ are eigenfunctions of the shift operator. The eigenvalues are complex phase factors which depend on the wave number indices (u, v) and the shift (k, l). When the shift is one wavelength, $(k, l) = (M/u, N/v)$, the phase factor reduces to 1 as we would expect.

Now we are curious to learn whether any linear shift-invariant operator has such a handy set of eigenimages. It turns out that all linear shift-invariant operators have the same set of eigenimages. We can prove this statement by referring to the *convolution theorem* (Section 2.3, Theorem 4, p. 52) which states that convolution is a point-wise multiplication in the Fourier space. Thus each element of the image representation in the Fourier space $\hat{g}_{uv}$ is multiplied by the complex scalar $\hat{h}_{uv}$. Each point in the Fourier space represents a base image, namely the complex exponential $^{uv}\boldsymbol{W}$ in Eq. (4.41) multiplied with the scalar $\hat{g}_{uv}$. Therefore, the complex exponentials are eigenfunctions of any convolution operator. The eigenvalues are then the elements of the transfer function, $\hat{H}_{uv}$. In conclusion, we can write

$$\mathcal{H}(\hat{g}_{uv}\,^{uv}\boldsymbol{W}) = \hat{h}_{uv}\hat{g}_{uv}\,^{uv}\boldsymbol{W}. \tag{4.43}$$

The fact that the eigenfunctions of LSI operators are the basis functions of the Fourier domain explains why convolution reduces to a multiplication in Fourier space and underlines the central importance of the Fourier transform for image processing.

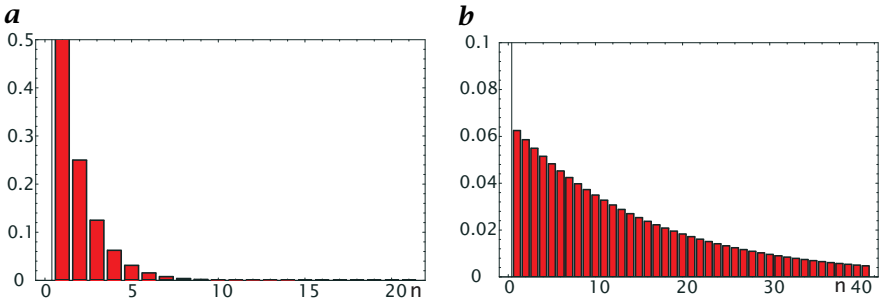

Figure 4.3: *Point spread function of the recursive filter* $g'_n = \alpha g'_{n-1} + (1-\alpha) g_n$ *for* ***a*** $\alpha = 1/2$ *and* ***b*** $\alpha = 15/16$.

4.3 Recursive Filters‡

4.3.1 Introduction‡

As convolution requires many operations, the question arises whether it is possible or even advantageous to include the already convolved neighboring gray values into the convolution at the next pixel. In this way, we might be able to do a convolution with fewer operations. In effect, we are able to perform convolutions with much less computational effort and also more flexibility. However, these filters, which are called *recursive filters*, are much more difficult to understand and to handle — especially in the multidimensional case.

For a first impression, we consider a very simple example. The simplest 1-D recursive filter we can think of has the general form

$$g'_n = \alpha g'_{n-1} + (1-\alpha) g_n. \tag{4.44}$$

This filter takes the fraction $1-\alpha$ from the previously calculated value and the fraction α from the current pixel. Recursive filters, in contrast to nonrecursive filters, work in a certain direction, in our example from left to right. For time series, the preferred direction seems natural, as the current state of a signal depends only on previous values. Filters that depend only on the previous values of the signal are called *causal filters*. For spatial data, however, no preferred direction exists. Consequently, we have to search for ways to construct filters with even and odd symmetry as they are required for image processing from recursive filters.

With recursive filters, the point spread function is no longer identical to the filter mask, but must be computed. From Eq. (4.44), we can calculate the *point spread function* or *impulse response* of the filter as the response of the filter to the *discrete delta function* (Section 4.2.5)

$$\delta_n = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}. \tag{4.45}$$

Recursively applying Eq. (4.44), we obtain

$$g'_{-1} = 0, \quad g'_0 = 1-\alpha, \quad g'_1 = (1-\alpha)\alpha, \quad \ldots, \quad g'_n = (1-\alpha)\alpha^n. \tag{4.46}$$

This equation shows three typical general properties of recursive filters:

- First, the impulse response is infinite (Fig. 4.3), despite the finite number of coefficients. For $|\alpha| < 1$ it decreases exponentially but never becomes exactly zero. In contrast, the impulse response of nonrecursive convolution filters is always finite. It is equal to the size of the filter mask. Therefore the two types of filters are sometimes named *finite impulse response filters* (*FIR filter*) and *infinite impulse response filters* (*IIR filter*).
- FIR filters are always *stable*. This means that the impulse response is finite. Then the response of a filter to any finite signal is finite. This is not the case for IIR filters. The stability of recursive filters depends on the filter coefficients. The filter in Eq. (4.44) is unstable for $|\alpha| > 1$, because then the impulse response diverges. In the simple case of Eq. (4.44) it is easy to recognize the instability of the filter. Generally, however, it is much more difficult to analyze the stability of a recursive filter, especially in two dimensions and higher.
- Any recursive filter can be replaced by a nonrecursive filter, in general with an infinite-sized mask. Its mask is given by the point spread function of the recursive filter. The inverse conclusion does not hold. This can be seen by the very fact that a non-recursive filter is always stable.

4.3.2 Transfer Function, z-Transform, and Stable Response‡

After this introductionary example, we are ready for a more formal discussion of *recursive filters*. Recursive filters include results from previous convolutions at neighboring pixels into the convolution sum and thus become directional. We discuss here only 1-D recursive filters. The general equation for a filter running from left to right is

$$g'_n = -\sum_{n''=1}^{S} a_{n''} g'_{n-n''} + \sum_{n'=-R}^{R} h_{n'} g_{n-n'}. \tag{4.47}$$

While the neighborhood of the nonrecursive part (coefficients h) is symmetric around the central point, the recursive part (coefficients a) uses only previously computed values. Such a recursive filter is called a *causal filter*.

If we put the recursive part on the left hand side of the equation, we observe that the recursive filter is equivalent to the following difference equation, also known as an *ARMA*(S,R) process (*autoregressive-moving average process*):

$$\sum_{n''=0}^{S} a_{n''} g'_{n-n''} = \sum_{n'=-R}^{R} h_{n'} g_{n-n'} \quad \text{with} \quad a_0 = 1. \tag{4.48}$$

The transfer function of such a filter with a recursive and a nonrecursive part can be computed by applying the *discrete Fourier transform* (Section 2.3.2) and making use of the shift theorem (Theorem 3, p. 52). Then

$$\hat{g}'(k) \sum_{n''=0}^{S} a_{n''} \exp(-2\pi i n'' k) = \hat{g}(k) \sum_{n'=-R}^{R} h_{n'} \exp(-2\pi i n' k). \tag{4.49}$$

Thus the *transfer function* is

$$\hat{h}(k) = \frac{\hat{g}'(k)}{\hat{g}(k)} = \frac{\sum_{n'=-R}^{R} h_{n'} \exp(-2\pi \mathrm{i} n' k)}{\sum_{n''=0}^{S} a_{n''} \exp(-2\pi \mathrm{i} n'' k)}. \tag{4.50}$$

The properties of the transfer function are governed by the zeros of the numerator and the denominator. Thus, a zero in the recursive part of the transfer function causes a zero in the transfer function, i. e., vanishing of the corresponding wave number. A zero in the recursive part causes a pole in the transfer function, i. e., an infinite response.

A determination of the zeros and thus a deeper analysis of the transfer function is not possible from Eq. (4.50). It requires an extension similar to the extension from real numbers to complex numbers that was used to introduce the Fourier transform (Section 2.3.2). We observe that the expressions for both the numerator and the denominator are polynomials in the *complex exponential* $\exp(2\pi \mathrm{i} k)$ of the form

$$\sum_{n=0}^{S} a_n \left(\exp(-2\pi \mathrm{i} k)\right)^n. \tag{4.51}$$

The complex exponential has a magnitude of one and thus covers the unit circle in the complex plane. The zeros of the polynomial must not be located one the unit circle but can be an arbitrary complex number. Therefore, it is useful to extend the polynomial so that it covers the whole complex plane. This is possible with the expression $z = r \exp(2\pi \mathrm{i} k)$ that describes a circle with the radius r in the complex plane.

With this extension we obtain a polynomial of the complex number z. As such we can apply the fundamental law of algebra that states that any polynomial of degree N can be factorized into N factors containing the roots or zeros of the polynomial:

$$\sum_{n=0}^{N} a_n z^n = a_n z^n \prod_{n=1}^{N} \left(1 - r_n z^{-1}\right). \tag{4.52}$$

With Eq. (4.52) we can factorize the recursive and nonrecursive parts of the polynomials in the transfer function into the following products:

$$\begin{aligned} \sum_{n=0}^{S} a_n z^{-n} &= z^{-S} \sum_{n=0}^{S} a_n z^{S-n} &&= z^{-S} \prod_{n=1}^{S} (1 - d_n z^{-1}), \\ \sum_{n=-R}^{R} h_n z^{-n} &= z^{-R} \sum_{n=0}^{S} h_n z^{R-n} &&= h_{-R} z^{-R} \prod_{n=1}^{2R} (1 - c_n z^{-1}). \end{aligned} \tag{4.53}$$

With $z = \exp(2\pi \mathrm{i} k)$ the transfer function can finally be written as

$$\hat{h}(z) = h_{-R} z^{S-R} \frac{\prod_{n'=1}^{2R} (1 - c_{n'} z^{-1})}{\prod_{n''=1}^{S} (1 - d_{n''} z^{-1})}. \tag{4.54}$$

Each of the factors $c_{n'}$ and $d_{n''}$ is a zero of the corresponding polynomial ($z = c_{n'}$ or $z = d_{n''}$).

The inclusion of the factor r in the extended transfer function results in an extension of the Fourier transform, the *z-transform*, which is defined as

$$\hat{g}(z) = \sum_{n=-\infty}^{\infty} g_n z^{-n}. \tag{4.55}$$

The z-transform of the series g_n can be regarded as the Fourier transform of the series $g_n r^{-n}$ [112]. The z-transform is the key mathematical tool to understand 1-D recursive filters. It is the discrete analogue to the *Laplace transform*. Detailed accounts of the z-transform are given by Oppenheim and Schafer [133] and Poularikas [141]; the 2-D z-transform is discussed by Lim [112].

Now we analyze the transfer function in more detail. The factorization of the transfer function is a significant advantage because each factor can be regarded as an individual filter. Thus each recursive filter can be decomposed into a cascade of simple recursive filters. As the factors are all of the form

$$f_n(\tilde{k}) = 1 - c_n \exp(-2\pi \mathrm{i}\tilde{k}) \tag{4.56}$$

and the impulse response of the filter must be real, the transfer function must be Hermitian, that is, $f(-k) = f^*(k)$. This can only be the case when either the zero c_n is real or a pair of factors exists with complex-conjugate zeros. This condition gives rise to two basic types of recursive filters, the *relaxation filter* and the *resonance filter* that are discussed in detail in Sections 4.3.5 and 4.3.6.

4.3.3 Higher-Dimensional Recursive Filters‡

Recursive filters can also be defined in higher dimensions with the same type of equation as in Eq. (4.47); also the transfer function and z-transform of higher-dimensional recursive filters can be written in the very same way as in Eq. (4.50). However, it is generally not possible to factorize the z-transform as in Eq. (4.54) [112]. From Eq. (4.54) we can immediately conclude that it will be possible to factorize a *separable* recursive filter because then the higher-dimensional polynomials can be factorized into 1-D polynomials. Given these inherent mathematical difficulties of higher-dimensional recursive filters, we will restrict the further discussion on 1-D recursive filters.

4.3.4 Symmetric Recursive Filtering‡

While a filter that uses only previous data is natural and useful for real-time processing of time series, it makes little sense for spatial data. There is no "before" and "after" in spatial data. Even worse is the signal-dependent spatial shift (delay) associated with recursive filters.

With a single recursive filter it is impossible to construct a so-called *zero-phase filter* with an even transfer function. Thus it is necessary to combine multiple recursive filters. The combination should either result in a zero-phase filter suitable for smoothing operations or a derivative filter that shifts the phase by 90°. Thus the transfer function should either be purely real or purely imaginary (Section 2.3.5).

We start with a 1-D causal recursive filter that has the transfer function

$$^{+}\hat{h}(\tilde{k}) = a(\tilde{k}) + \mathrm{i}b(\tilde{k}). \tag{4.57}$$

The superscript "+" denotes that the filter runs in positive coordinate direction. The transfer function of the same filter but running in the opposite direction has a similar transfer function. We replace $\tilde{k}$ by $-\tilde{k}$ and note that $a(-\tilde{k}) = a(+\tilde{k})$ and $b(-\tilde{k}) = -b(\tilde{k})$), because the transfer function of a real PSF is Hermitian (Section 2.3.5), and obtain

$$^{-}\hat{h}(\tilde{k}) = a(\tilde{k}) - \mathrm{i}b(\tilde{k}). \tag{4.58}$$

Thus, only the sign of the imaginary part of the transfer function changes when the filter direction is reversed.

We now have three possibilities to combine the two transfer functions (Eqs. (4.57) and (4.58)) either into a purely real or imaginary transfer function:

$$\begin{aligned}
&\text{Addition} && ^{e}\hat{h}(\tilde{k}) = \frac{1}{2}\left(^{+}\hat{h}(\tilde{k}) + {}^{-}\hat{h}(\tilde{k})\right) = a(\tilde{k}),\\
&\text{Subtraction} && ^{o}\hat{h}(\tilde{k}) = \frac{1}{2}\left(^{+}\hat{h}(\tilde{k}) - {}^{-}\hat{h}(\tilde{k})\right) = \mathrm{i}b(\tilde{k}),\\
&\text{Multiplication} && \hat{h}(\tilde{k}) = {}^{+}\hat{h}(\tilde{k})\,{}^{-}\hat{h}(\tilde{k}) = a^2(\tilde{k}) + b^2(\tilde{k}).
\end{aligned} \tag{4.59}$$

Addition and multiplication (consecutive application) of the left and right running filter yields filters of even symmetry and a real transfer function, while subtraction results in a filter of odd symmetry and a purely imaginary transfer function.

4.3.5 Relaxation Filters‡

The simple recursive filter discussed in Section 4.3.1

$$g'_n = a_1 g'_{n\mp1} + h_0 g_n \quad \text{with} \quad a_1 = \alpha,\ h_0 = (1-\alpha) \tag{4.60}$$

and the point spread function

$$^{\pm}r_{\pm n} = \begin{cases} (1-\alpha)\alpha^n & n \geq 0 \\ 0 & \text{else} \end{cases} \tag{4.61}$$

is a *relaxation filter*. The transfer function of the filter runnning either in forward or in backward direction is, according to Eq. (4.50) with Eq. (4.60), given by

$$^{\pm}\hat{r}(\tilde{k}) = \frac{1-\alpha}{1-\alpha\exp(\mp\pi\mathrm{i}\tilde{k})} \quad \text{with} \quad \alpha \in \mathbb{R}. \tag{4.62}$$

The transfer function Eq. (4.62) is complex and can be divided into its real and imaginary parts as

$$^{\pm}\hat{r}(\tilde{k}) = \frac{1-\alpha}{1-2\alpha\cos\pi\tilde{k}+\alpha^2}\left[(1-\alpha\cos\pi\tilde{k}) \mp \mathrm{i}\alpha\sin\pi\tilde{k}\right]. \tag{4.63}$$

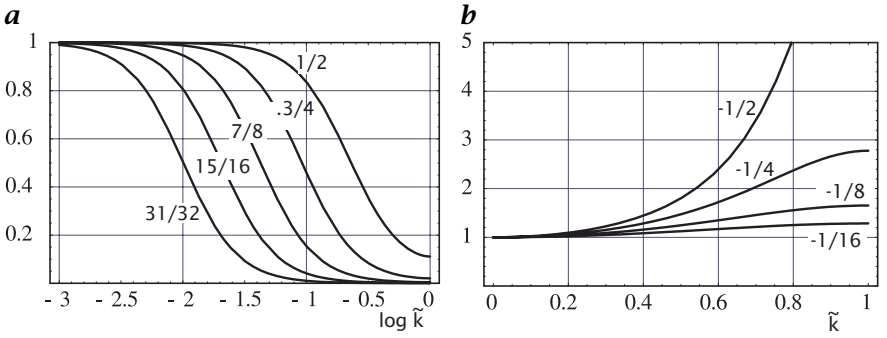

Figure 4.4: *Transfer function of the relaxation filter $g'_n = \alpha g'_{n\mp1} + (1-\alpha)g_n$ applied first in forward and then in backward direction for* ***a*** *positive; and* ***b*** *negative values of α as indicated.*

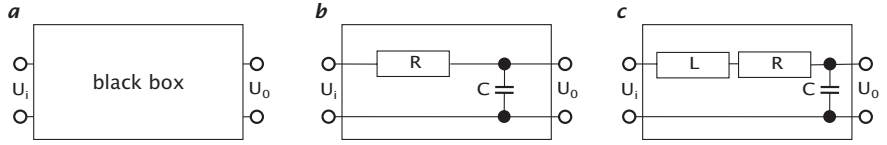

Figure 4.5: *Analog filter for time series.* ***a*** *Black-box model: a signal U_i is put into an unknown system and at the output we measure the signal U_o.* ***b*** *A resistor-capacitor circuit as a simple example of an analog lowpass filter.* ***c*** *Damped resonance filter consisting of an inductor L, a resistor R, and a capacitor C.*

After Eq. (4.59), we can then compute the transfer function $\hat{r}$ for the resulting symmetric filter if we apply the relaxation filters successively in positive and negative direction:

$$\hat{r}(\tilde{k}) = \frac{(1-\alpha)^2}{1-2\alpha\cos\pi\tilde{k}+\alpha^2} = \frac{1}{1+\beta-\beta\cos\pi\tilde{k}} \tag{4.64}$$

with

$$\beta = \frac{2\alpha}{(1-\alpha)^2} \quad \text{and} \quad \alpha = \frac{1+\beta-\sqrt{1+2\beta}}{\beta}$$

From Eq. (4.61) we can conclude that the relaxation filter is stable if $|\alpha| < 1$, which corresponds to $\beta \in]-1/2, \infty[$. As already noted, the transfer function is one for small wave numbers. A Taylor series in $\tilde{k}$ results in

$$\hat{r}(\tilde{k}) \approx= 1 - \frac{\alpha}{(1-\alpha)^2}(\pi\tilde{k})^2 + \frac{\alpha((1+10\alpha+\alpha^2)}{12(1-\alpha^2)^2}(\pi\tilde{k})^4. \tag{4.65}$$

If α is positive, the filter is a low-pass filter (Fig. 4.4a). It can be tuned by adjusting α. If α is approaching 1, the averaging distance becomes infinite. For negative α, the filter enhances high wave numbers (Fig. 4.4b).

This filter is the discrete analog to the first-order differential equation $\dot{y} + \tau y = 0$ describing a relaxation process with the relaxation time $\tau = -\Delta t / \ln\alpha$.

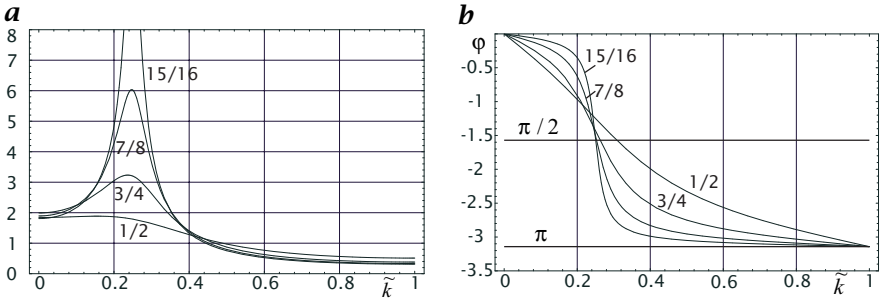

Figure 4.6: ***a*** *Magnitude and* ***b*** *phase shift of the transfer function of the resonance filter according to Eq. (4.67) for* $\tilde{k}_0 = 1/4$ *and values for* r *as indicated.*

An example is the simple resistor-capacitor circuit shown in Fig. 4.5b. The differential equation for this filter can be derived from Kirchhoff's current-sum law. The current flowing through the resistor from U_i to U_o must be equal to the current flowing into the capacitor. Since the current flowing into a capacitor is proportional to the temporal derivative of the potential U_o, we end up with the first-order differential equation

$$\frac{U_i - U_o}{R} = C\frac{\partial U_o}{\partial t}. \tag{4.66}$$

and the time constant is given by $\tau = RC$.

4.3.6 Resonance Filters‡

The second basic type of a recursive filter that we found from the discussion of the transfer function in Section 4.3.2 has a pair of complex-conjugate zeros. Therefore, the transfer function of this filter running in forward or backward direction is

$$\begin{aligned} {}^{\pm}\hat{s}(\tilde{k}) &= \frac{1}{(1 - r\exp(\mathrm{i}\pi\tilde{k}_0)\exp(\mp\mathrm{i}\pi\tilde{k}))(1 - r\exp(-\mathrm{i}\pi\tilde{k}_0)\exp(\mp\mathrm{i}\pi\tilde{k}))} \\ &= \frac{1}{1 - 2r\cos(\pi\tilde{k}_0)\exp(\mp\mathrm{i}\pi\tilde{k}) + r^2\exp(\mp 2\mathrm{i}\pi\tilde{k})}. \end{aligned} \tag{4.67}$$

The second row of the equation shows that this recursive filter has the coefficients $h_0 = 1$, $a_1 = -2r\cos(\pi\tilde{k}_0)$, and $a_2 = r^2$ hat:

$$g'_n = g_n + 2r\cos(\pi\tilde{k}_0)g'_{n\mp 1} - r^2 g'_{n\mp 2}. \tag{4.68}$$

From the transfer function in Eq. (4.67) we conclude that this filter is a *bandpass filter* with a passband wave number of $\pm\tilde{k}_0$ (Fig. 4.6). For $r = 1$ the transfer function has two poles at $\tilde{k} = \pm\tilde{k}_0$.

The impulse response of this filter is after [133]

$$h_{\pm n} = \begin{cases} \dfrac{r^n}{\sin\pi\tilde{k}_0}\sin[(n+1)\pi\tilde{k}_0] & n \geq 0 \\ 0 & n < 0 \end{cases}. \tag{4.69}$$

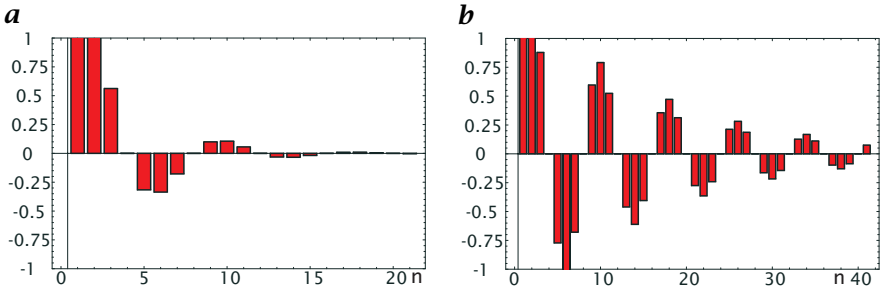

Figure 4.7: *Point spread function of the recursive resonance filter according to Eq. (4.68) for* ***a*** $\tilde{k}_0 = 1/4$, $r = 3/4$ *and* ***b*** $\tilde{k}_0 = 1/4$, $r = 15/16$.

This means that the filter acts as a damped oscillator. The parameter $\tilde{k}_0$ gives the wave number of the oscillation and the parameter r is the damping constant (Fig. 4.7). The filter is only stable if $r \leq 1$.

If we run the filter back and forth, the resulting filter has a real transfer function $\hat{s}(\tilde{k}) = {}^{+}\hat{s}(\tilde{k})\,{}^{-}\hat{s}(\tilde{k})$ that is given by

$$\hat{s}(\tilde{k}) = \frac{1}{\left(1 - 2r\cos[\pi(\tilde{k} - \tilde{k}_0)] + r^2\right)\left(1 - 2r\cos[\pi(\tilde{k} + \tilde{k}_0)] + r^2\right)}. \tag{4.70}$$

The transfer function of this filter can be normalized so that its maximal value becomes 1 in the passband by setting the nonrecursive filter coefficient h_0 to $(1 - r^2)\sin(\pi\tilde{k}_0)$. Then we obtain the following modified recursion

$$g'_n = (1 - r^2)\sin(\pi\tilde{k}_0)g_n + 2r\cos(\pi\tilde{k}_0)g'_{n\mp1} - r^2 g'_{n\mp2}. \tag{4.71}$$

For symmetry reasons, the factors become most simple for a resonance wave number of $\tilde{k}_0 = 1/2$. Then the recursive filter is

$$g'_n = (1 - r^2)g_n - r^2 g'_{n\mp2} = g_n - r^2(g_n + g'_{n\mp2}) \tag{4.72}$$

with the transfer function

$$\hat{s}(\tilde{k}) = \frac{(1 - r^2)^2}{1 + r^4 + 2r^2\cos(2\pi\tilde{k})}. \tag{4.73}$$

The maximum response of this filter at $\tilde{k} = 1/2$ is one and the minimum response at $\tilde{k} = 0$ and $\tilde{k} = 1$ is $((1 - r^2)/(1 + r^2))^2$.

This resonance filter is the discrete analog to a linear system governed by the second-order differential equation $\ddot{y} + 2\tau\dot{y} + \omega_0^2 y = 0$, the damped harmonic oscillator such as the LRC circuit in Fig. 4.5c. The circular eigenfrequency ω_0 and the time constant τ of a real-world oscillator are related to the parameters of the discrete oscillator, r and $\tilde{k}_0$ by [81]

$$r = \exp(-\Delta t/\tau) \quad \text{and} \quad \tilde{k}_0 = \omega_0 \Delta t/\pi. \tag{4.74}$$

4.3.7 LSI Filters and System Theory‡

The last example of the damped oscillator illustrates that there is a close relationship between discrete filter operations and analog physical systems. Thus, digital filters model a real-world physical process. They pattern how the corresponding system would respond to a given input signal g. Actually, we will make use of this equivalence in our discussion of image formation in Chapter 7. There we will find that imaging with a homogeneous optical system is completely described by its point spread function and that the image formation process can be described by convolution. Optical imaging together with physical systems such as electrical filters and oscillators of all kinds can thus be regarded as representing an abstract type of process or system, called a *linear shift-invariant system* or short *LSI*.

This generalization is very useful for image processing, as we can describe both image formation and image processing as convolution operations with the same formalism. Moreover, the images observed may originate from a physical process that can be modeled by a linear shift-invariant system. Then the method for finding out how the system works can be illustrated using the black-box model (Fig. 4.5a). The black box means that we do not know the composition of the system observed or, physically speaking, the laws that govern it. We can find them out by probing the system with certain signals (input signals) and watching the response by measuring some other signals (output signals). If it turns out that the system is linear, it will be described completely by the impulse response.

Many biological and medical experiments are performed in this way. Biological systems are typically so complex that the researchers often stimulate them with signals and watch for responses in order to find out how they work and to construct a model. From this model more detailed research may start to investigate how the observed system functions might be realized. In this way many properties of biological visual systems have been discovered. But be careful — a model is not the reality! It pictures only the aspect that we probed with the applied signals.

4.4 Rank Value Filtering

The considerations on how to combine pixels have resulted in the powerful concept of linear shift-invariant systems. Thus we might be tempted to think that we have learnt all we need to know for this type of image processing operation. This is not the case. There is another class of operations which works on a quite different principle.

We might characterize a convolution with a filter mask by weighting and summing up. The class of operations to combine neighboring pixels we are considering now is characterized by comparing and selecting. Such a filter is called a *rank-value filter*. For this we take all the gray values of the pixels which lie within the filter mask and sort them by ascending gray value. This sorting is common to all rank value filters. They only differ by the position in the list from which the gray value

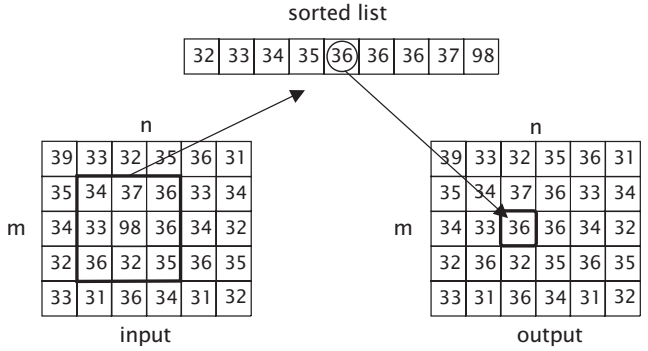

Figure 4.8: *Illustration of the principle of rank value filters with a* 3×3 *median filter.*

is picked out and written back to the center pixel. The filter operation which selects the medium value is called the *median filter*. Figure 4.8 illustrates how the median filter works. The filters choosing the minimum and maximum values are denoted as the *minimum* and *maximum filter*, respectively.

The median filter is a nonlinear operator. For the sake of simplicity, we consider a one-dimensional case with a 3-element median filter. It is easy to find two vectors for which the median filter is not linear:

$$\begin{aligned} \mathcal{M}([\cdots 0\,1\,0\,0 \cdots] + [\cdots 0\,0\,1\,0 \cdots]) &= [\cdots 0\,1\,1\,0 \cdots] \neq \\ \mathcal{M}[\cdots 0\,1\,0\,0 \cdots] + \mathcal{M}[\cdots 0\,0\,1\,0 \cdots] &= [\cdots 0\,0\,0\,0 \cdots]. \end{aligned}$$

There are a number of significant differences between convolution filters and rank value filters. Most important, rank value filters belong to the class of *nonlinear filters*. Consequently, it is much more difficult to understand their general properties. As rank value filters do not perform arithmetic operations but select pixels, we will never run into rounding problems. These filters map a discrete set of gray values onto themselves.

4.5 Further Readings‡

The classical concepts of filtering of discrete time series, especially recursive filters and the z transform are discussed in Oppenheim and Schafer [133] and Proakis and Manolakis [144], 2-D filtering in Lim [112]. A detailed account of nonlinear filters, especially median filters, is given by Huang [75] and Pitas and Venetsanopoulos [140].

5 Multiscale Representation

5.1 Scale

5.1.1 Introduction

The neighborhood operations discussed in Chapter 4 can only be the starting point for image analysis. This class of operators can only extract local features at scales of at most a few pixels distance. It is obvious that images contain information also at larger scales. To extract object features at these larger scales, we need correspondingly larger filter masks. The use of large masks, however, results in a significant increase in computational costs. If we use a mask of size R^W in a W-dimensional image the number of operations is proportional to R^W. Thus a doubling of the scale leads to a four- and eight-fold increase in the number of operations in 2- and 3-dimensional images, respectively. For a ten times larger scale, the number of computations increases by a factor of 100 and 1000 for 2- and 3-dimensional images, respectively.

The explosion in computational cost is only the superficial expression of a problem with deeper roots. We illustrate it with a simple task, the detection of edges and lines at different resolutions. To this end, we use the same image row but blur it by different degrees (Fig. 5.1). We define the corresponding scale as the distance over which the image has been blurred and analyze the gray value differences over this distance.

We first investigate gray value differences at high resolution, a scale of just one pixel distance (Fig. 5.1a, b). At this fine scale, the change in gray values is dominated by the noisy background of the image. Any detection of gray value changes caused by the contrast between objects and background is inaccurate and erroneous. The problem is caused by a *scale mismatch*: the gray values only vary on larger scales than the operators used to detect them.

If we take instead a low resolution (Fig. 5.1e, f), the lines are blurred so much that the contrast has significantly decreased. Moreover, two closely spaced lines in the left part of the signal have merged into one object at this coarse resolution. Therefore the detection of edges and lines is suboptimal again. At a resolution comparable to the line width, however, the line detection seems to be optimal (Fig. 5.1c, d). Noise is significantly reduced compared to the finest scale (Fig. 5.1a) but the

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2

Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

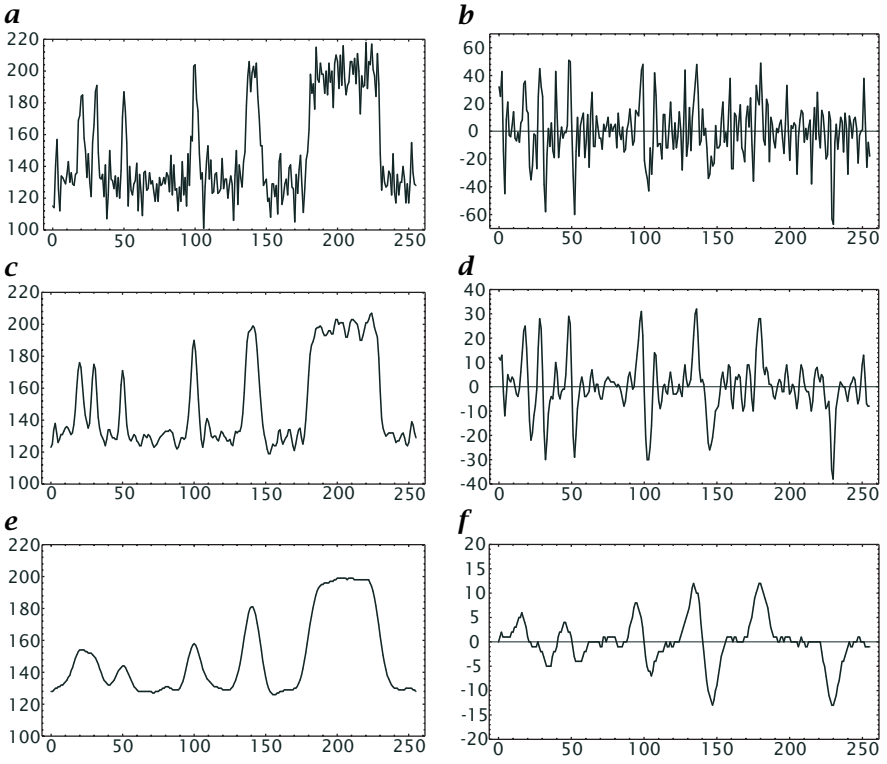

Figure 5.1: *Lines and edges at* ***a*** *high,* ***c*** *medium, and* ***e*** *low resolution.* ***b****,* ***d****, and* ***f*** *Subtraction of neighboring pixels for edge detection for* ***a****,* ***c****, and* ***e****, respectively.*

contrast between the line and the background is not yet diminished as in Fig. 5.1e.

From the discussion of this example we can conclude that the detection of certain features in an image is optimal at a certain scale. This scale depends, of course, on the characteristic scales contained in the object to be detected. Optimal processing of an image thus requires the representation of an image at different scales. In order to meet this demand, we need a *multiscale representation* of images. In this chapter, we will first illuminate the relation between the spatial and wave number representation of images under this perspective (Section 5.1.2). Then we will introduce the *scale space* (Section 5.2), discuss how it can be generated by a diffusion process, and describe its basic properties. Finally, in Section 5.3 we will turn to efficient *multigrid representations* such as the *Gaussian pyramid* (Section 5.3.2) and the *Laplacian pyramid* (Section 5.3.3).

5.1.2 Spatial Versus Wave Number Representation

In Chapter 2 we discussed in detail the representation of images in the spatial and wave number domain. In this section we will revisit both representations under the perspective of how to generate a multiscale representation of an image.

If we represent an image on a grid in the spatial domain, we do not have any information at all about the wave numbers contained at that point in the image. We know the position with an accuracy of the grid constant Δx, but the local wave number at this position may be anywhere in the range of the possible wave numbers from 0 to $M\Delta k = 2\pi M/\Delta x$.

In the wave number representation, we have the reverse case. Each pixel in this domain represents one wave number with the highest wave number resolution possible for the given image size. But any positional information is lost, as one point in the wave number space represents a periodic structure that is spread over the whole image.

The above discussion shows that the representation of an image in either the spatial or wave number domain constitute two opposite extremes. We can optimize either the spatial or the wave number resolution but the resolution in the other domain is completely lost. What we need for a multiscale image representation is a type of joint resolution that allows for a separation into different wave number ranges (scales) but still preserves as much spatial resolution as possible.

5.1.3 Windowed Fourier Transform

One way to approach a joint space-wave number representation is the *windowed Fourier transform*. As the name says, the Fourier transform is not applied to the whole image but only to a section of the image that is formed by multiplying the image with a window function $w(\boldsymbol{x})$. The window function has a maximum at $\boldsymbol{x} = \boldsymbol{0}$ and decreases monotonically with $|\boldsymbol{x}|$ towards zero. The maximum of the window function is then put at each point $\boldsymbol{x}$ of the image to compute a windowed Fourier transform for each point:

$$\hat{g}(\boldsymbol{x}, \boldsymbol{k}_0) = \int_{-\infty}^{\infty} g(\boldsymbol{x}') w(\boldsymbol{x}' - \boldsymbol{x}) \exp\left(-2\pi \mathrm{i} \boldsymbol{k}_0 \boldsymbol{x}'\right) \mathrm{d}x'^2. \tag{5.1}$$

The integral in Eq. (5.1) almost looks like a convolution integral (Eq. (2.56), ≻ R4). To convert it into a convolution integral we observe that $w(-\boldsymbol{k}) = w(\boldsymbol{k})$ and rearrange the second part of Eq. (5.1):

$$\begin{aligned} & w(\boldsymbol{x}' - \boldsymbol{x}) \exp\left(-2\pi \mathrm{i} \boldsymbol{k}_0 \boldsymbol{x}'\right) \\ = \quad & w(\boldsymbol{x} - \boldsymbol{x}') \exp\left(2\pi \mathrm{i} \boldsymbol{k}_0 (\boldsymbol{x} - \boldsymbol{x}')\right) \exp\left(-2\pi \mathrm{i} \boldsymbol{k}_0 \boldsymbol{x}\right). \end{aligned} \tag{5.2}$$

Then we can write Eq. (5.1) as a convolution

$$\hat{g}(\boldsymbol{x}, \boldsymbol{k}_0) = (g(\boldsymbol{x}) * w(\boldsymbol{x}) \exp(2\pi \mathrm{i} \boldsymbol{k}_0 \boldsymbol{x})) \exp(-2\pi \mathrm{i} \boldsymbol{k}_0 \boldsymbol{x}). \tag{5.3}$$

This means that the local Fourier transform corresponds to a convolution with the complex convolution kernel $w(\boldsymbol{x}) \exp(2\pi \mathrm{i} \boldsymbol{k}_0 \boldsymbol{x})$ except for a phase factor $\exp(-2\pi \mathrm{i} \boldsymbol{k}_0 \boldsymbol{x})$. Using the *shift theorem* (Theorem 3, p. 52, ≻ R4), the transfer function of the convolution kernel can be computed to be

$$w(\boldsymbol{x}) \exp(2\pi \mathrm{i} \boldsymbol{k}_0 \boldsymbol{x}) \circ\!\!-\!\!\bullet\, \hat{w}(\boldsymbol{k} - \boldsymbol{k}_0). \tag{5.4}$$

This means that the convolution kernel $w(\boldsymbol{x}) \exp(2\pi \mathrm{i} \boldsymbol{k}_0 \boldsymbol{x})$ is a *bandpass filter* with a peak wave number of $\boldsymbol{k}_0$. The width of the bandpass is inversely proportional to the width of the window function. In this way, the spatial and wave number resolutions are interrelated to each other. As an example, we take a Gaussian window function

$$\exp\left(-\frac{x^2}{2\sigma_x^2}\right). \tag{5.5}$$

Its Fourier transform (≻ R4, ≻ R5), is

$$\frac{1}{\sqrt{2\pi}\sigma_x} \exp\left(-2\pi^2 k^2 \sigma_x^2\right). \tag{5.6}$$

Consequently, the product of the standard deviations in the space and wave number domain ($\sigma_k^2 = 1/(4\pi\sigma_x^2)$) is a constant: $\sigma_x^2 \sigma_k^2 = 1/(4\pi)$). This fact establishes the classical *uncertainty relation* (Theorem 7, p. 55). It states that the product of the standard deviations of any Fourier transform pair is larger than or equal to $1/(4\pi)$. As the Gaussian window function reaches the theoretical minimum it is an optimal choice; a better wave number resolution cannot be achieved with a given spatial resolution.

5.2 Scale Space†

As we have seen with the example of the windowed Fourier transform in the previous section, the introduction of a characteristic *scale* adds a new coordinate to the representation of image data. Besides the spatial resolution, we have a new parameter that characterizes the current resolution level of the image data. The scale parameter is denoted by ξ. A data structure that consists of a sequence of images with different resolutions is known as a *scale space*; we write $g(\boldsymbol{x}, \xi)$ to indicate the scale space of the image $g(\boldsymbol{x})$.

Next, in Section 5.2.1, we discuss a physical process, diffusion, that is suitable for generating a scale space. Then we discuss the general properties of a scale space in Section 5.2.2.

5.2.1 Scale Generation by Diffusion

The generation of a scale space requires a process that can blur images to a controllable degree. Diffusion is a transport process that tends to level out concentration differences [23]. In physics, diffusion processes govern the transport of heat, matter, and momentum leading to an ever increasing equalization of spatial concentration differences. If we identify the time with the scale parameter ξ, the diffusion process establishes a scale space.

To apply a diffusion process to an image, we regard the gray value g as the concentration of a chemical species. The elementary law of diffusion states that the flux density $\boldsymbol{j}$ is directed against the concentration gradient ∇g and proportional to it:

$$\boldsymbol{j} = -D\nabla g \tag{5.7}$$

where the constant D is known as the *diffusion coefficient*. Using the continuity equation

$$\frac{\partial g}{\partial t} + \nabla \boldsymbol{j} = 0 \tag{5.8}$$

the diffusion equation is

$$\frac{\partial g}{\partial t} = \nabla(D\nabla g). \tag{5.9}$$

For the case of a homogeneous diffusion process (D does not depend on the position), the equation reduces to

$$\frac{\partial g}{\partial t} = D\Delta g \tag{5.10}$$

where

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{5.11}$$

is the *Laplacian operator*. It is easy to show that the general solution to this equation is equivalent to a convolution with a smoothing mask. To this end, we perform a spatial Fourier transform which results in

$$\frac{\partial \hat{g}(\boldsymbol{k})}{\partial t} = -4\pi^2 D|\boldsymbol{k}|^2 \hat{g}(\boldsymbol{k}) \tag{5.12}$$

reducing the equation to a linear first-order differential equation with the general solution

$$\hat{g}(\boldsymbol{k}, t) = \exp(-4\pi^2 D|\boldsymbol{k}|^2 t)\hat{g}(\boldsymbol{k}, 0), \tag{5.13}$$

where $\hat{g}(\boldsymbol{k}, 0)$ is the Fourier transformed image at time zero.

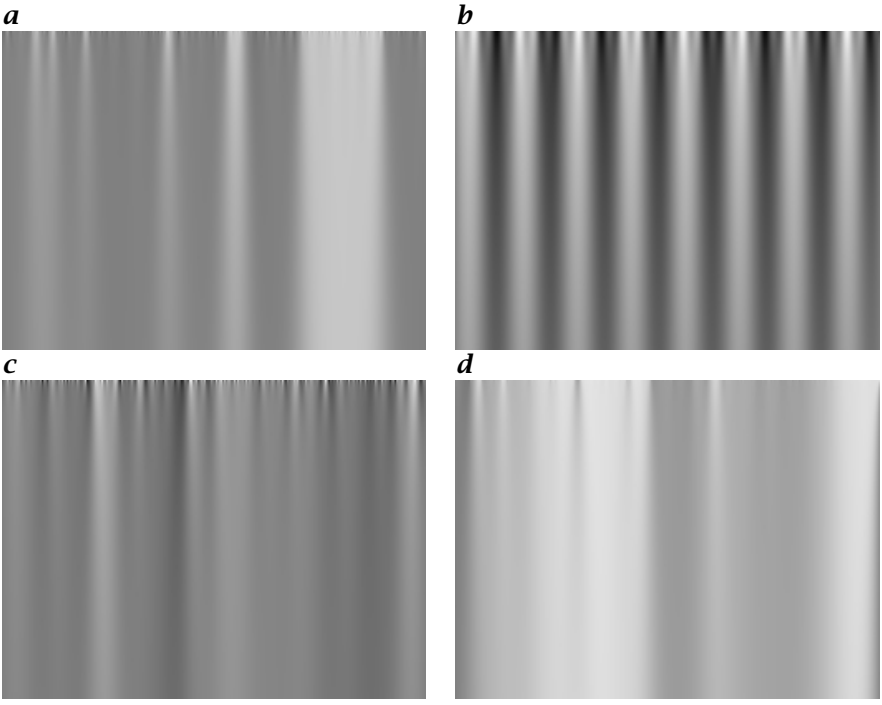

Figure 5.2: *Scale space of some one-dimensional signals:* ***a*** *edges and lines;* ***b*** *a periodic pattern;* ***c*** *a random signal;* ***d*** *row 10 from the image shown in Fig. 11.6a. The vertical coordinate is the scale parameter ξ.*

Multiplication of the image in the Fourier space with the Gaussian function in Eq. (5.13) is equivalent to a convolution with the same function but of reciprocal width. Thus,

$$g(\boldsymbol{x},t) = \frac{1}{2\pi\sigma^2(t)} \exp\left(-\frac{|\boldsymbol{x}|^2}{2\sigma^2(t)}\right) * g(\boldsymbol{x},0) \tag{5.14}$$

with

$$\sigma(t) = \sqrt{2Dt}. \tag{5.15}$$

Equation Eq. (5.15) shows that the degree of smoothing expressed by the standard deviation σ increases only by the power half with time. Therefore we set the scale parameter ξ equal to the square of the standard deviation:

$$\xi = 2Dt. \tag{5.16}$$

It is important to note that this formulation of the scale space is valid for images of any dimension. It could also be extended to image sequences. The scale parameter is not identical to the time although we used a physical diffusion process that proceeds with time to derive it.

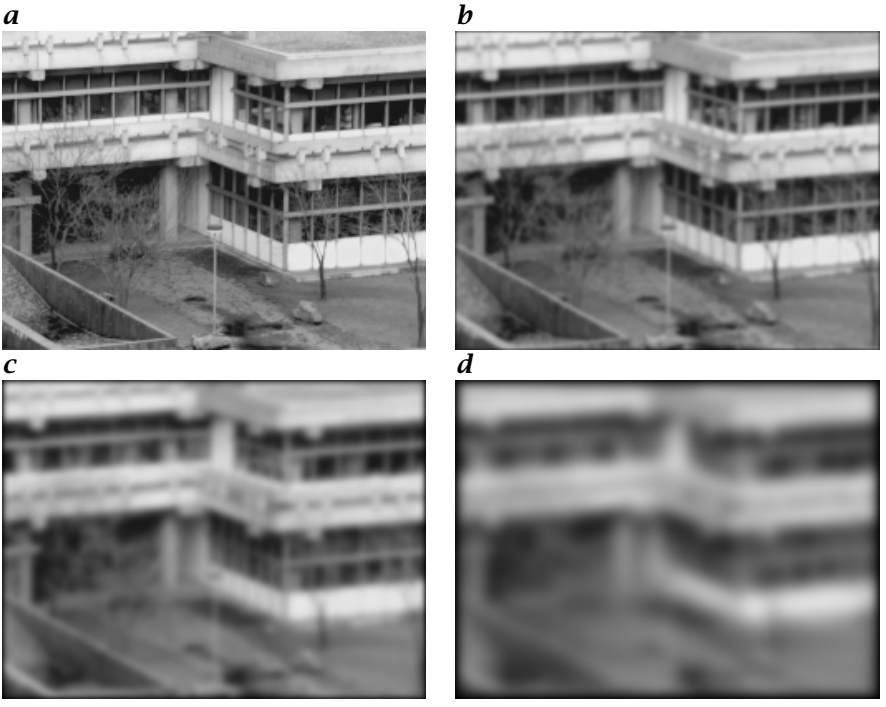

Figure 5.3: *Scale space of a two-dimensional image:* ***a*** *original image;* ***b***, ***c***, *and* ***d*** *at scale parameters σ 1, 2, and 4, respectively.*

If we compute a scale space representation of an image sequence, it is useful to scale the time coordinate with a characteristic velocity u_0 so that it has the same dimension as the spatial coordinates:

$$t' = u_0 t. \tag{5.17}$$

We add this coordinate to the spatial coordinates and get a new coordinate vector

$$\boldsymbol{x} = [x_1, x_2, u_0 t]^T \quad \text{or} \quad \boldsymbol{x} = [x_1, x_2, x_3, u_0 t]^T. \tag{5.18}$$

In the same way, we extend the wave number vector by a scaled frequency:

$$\boldsymbol{k} = [k_1, k_2, \omega/u_0]^T \quad \text{or} \quad \boldsymbol{k} = [k_1, k_2, k_3, \omega/u_0]^T. \tag{5.19}$$

With Eqs. (5.18) and (5.19) all equations derived above, e. g., Eqs. (5.13) and (5.14), can also be applied to scale spaces of image sequences. For discrete spaces, of course, no such scaling is required. It is automatically fixed by the spatial and temporal sampling intervals: $u_0 = \Delta x/\Delta t$.

As an illustration, Fig. 5.2 shows the scale space of some characteristic one-dimensional signals: noisy edges and lines, a periodic pattern, a random signal, and a row of an image. These examples nicely demonstrate a general property of scale spaces. With increasing scale parameter ξ, the signals become increasingly blurred, more and more details are lost. This feature can be most easily seen by the transfer function of the scale space representation in Eq. (5.13). The transfer function is always positive and monotonically decreasing with the increasing scale parameter ξ for all wave numbers. This means that no structure is amplified. All structures are attenuated with increasing ξ, and smaller structures always faster than coarser structures. In the limit of $\xi \to \infty$ the scale space converges to a constant image with the mean gray value. A certain feature exists only over a certain scale range. In Fig. 5.2a we can observe that edges and lines disappear and two objects merge into one.

For two-dimensional images, a continuous representation of the scale space would give a three-dimensional data structure. Therefore Fig. 5.3 shows individual images for different scale parameters ξ as indicated.

5.2.2 General Properties of a Scale Space

In this section, we discuss some general properties of scale spaces. More specifically, we want to know what kinds of condition must be met by a filter kernel generating a scale space. We will discuss two basic requirements. First, no new details must be added with increasing scale parameter. From the perspective of information theory, we may say that the information content in the signal should continuously decrease with the scale parameter.

The second property is related to the general principle of *scale invariance*. This basically means that we can start smoothing the signal at any scale parameter in the scale space and still obtain the same scale space. Here, we will give only some basic ideas about these elementary properties and no proofs. For a detailed treatment of the scale space theory we refer to the recent monograph on linear scale space theory by Lindeberg [113].

The linear homogenous and isotropic diffusion process has according to Eq. (5.14) the convolution kernel

$$\boldsymbol{B}(\boldsymbol{x}, \xi) = \frac{1}{2\pi\xi} \exp\left(-\frac{|\boldsymbol{x}|^2}{2\xi}\right) \tag{5.20}$$

and the transfer function Eq. (5.13)

$$\hat{\boldsymbol{B}}(\boldsymbol{k}, \xi) = \exp(-2\pi^2 |\boldsymbol{k}|^2 \xi). \tag{5.21}$$

In these equations, we have replaced the explicit dependence on time by the scale parameter ξ using Eq. (5.16). In a representation-independent

way, we denote the scale space generating operator as

$$\mathcal{B}(\xi). \tag{5.22}$$

The information-decreasing property of the scale space with ξ can be formulated mathematically in different ways. We express it here with the *minimum-maximum principle* which states that local extrema must not be enhanced. This means that the gray value at a local maximum or minimum must not increase or decrease, respectively. For a diffusion process this is an intuitive property. For example, in a heat transfer problem, a hot spot must not become hotter or a cool spot cooler. The Gaussian kernel Eq. (5.20) meets the minimum-maximum principle.

The second important property of the scale space is related to the *scale invariance* principle. We want to start the generating process at any scale parameter and still get the same scale space. More quantitatively, we can formulate this property as

$$\mathcal{B}(\xi_2)\mathcal{B}(\xi_1) = \mathcal{B}(\xi_1 + \xi_2). \tag{5.23}$$

This means that the smoothing of the scale space at the scale ξ_1 by an operator with the scale ξ_2 is equivalent to the application of the scale space operator with the scale $\xi_1 + \xi_2$ to the original image. Alternatively, we can state that the representation at the coarser level ξ_2 can be computed from the representation at the finer level ξ_1 by applying

$$\mathcal{B}(\xi_2) = \mathcal{B}(\xi_2 - \xi_1)\mathcal{B}(\xi_1) \quad \text{with} \quad \xi_2 > \xi_1. \tag{5.24}$$

From Eqs. (5.20) and (5.21) we can easily verify that Eqs. (5.23) and (5.24) are true. In mathematics the properties Eqs. (5.23) and (5.24) are referred to as the *semi-group property*.

Conversely, we can ask what scale space generating kernels exist that meet both the minimum-maximum principle and the semi-group property. The answer to this question may be surprising. The Gaussian kernel is the *only* convolution kernel that meets both these criteria and is in addition isotropic and homogeneous [113]. This feature puts the Gaussian convolution kernel and — as we will see later — its discrete counterpart the binomial kernel into a unique position for image processing. It will be elaborated in more detail in Section 11.4.

It is always instructive to discuss a counterexample. The most straightforward smoothing kernel for a W-dimensional image — known as the *moving average* — is the box filter

$$\boldsymbol{R}(\boldsymbol{x}, \xi) = \frac{1}{\xi^W} \prod_{w=1}^{W} \Pi\left(\frac{x_w}{\xi}\right) \tag{5.25}$$

with the transfer function

$$\hat{\boldsymbol{R}}(\boldsymbol{k}, \xi) = \prod_{w=1}^{W} \frac{\sin(k_w \xi/2)}{k_w \xi/2}. \tag{5.26}$$

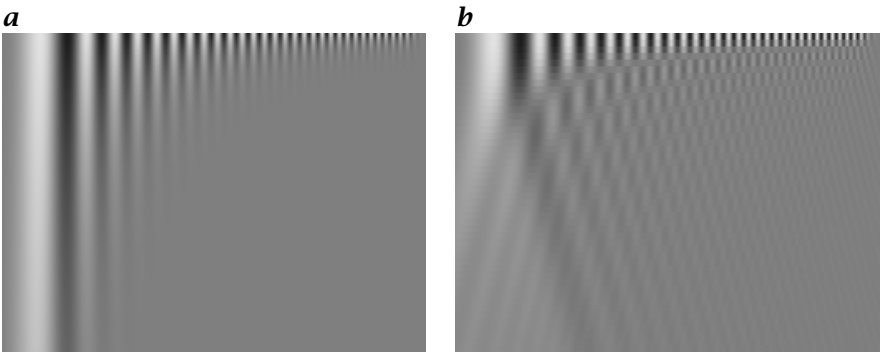

Figure 5.4: *Scale space of a 1-D signal with varying wave number computed with a* ***a*** *Gaussian and* ***b*** *box kernel. The scale parameter runs from top to bottom.*

This kernel meets neither the minimum-maximum principle nor the semigroup property. Figure 5.4 compares scale spaces of a periodic signal with varying wave number generated with a Gaussian and a box kernel. In Fig. 5.4b it becomes evident that the box kernel does not meet the minimum-maximum principle as structures decrease until they are completely removed but then appear again.

5.2.3 Quadratic and Logarithmic Scale Spaces‡

Despite the mathematical beauty of scale space generation with a Gaussian convolution kernel, this approach has one significant disadvantage. The standard deviation of the smoothing increases only by the power half with time Eq. (5.15). Therefore the scale parameter ξ is only proportional to the square of the standard deviation. This results in a nonlinear scale coordinate. While smoothing goes fast for fine scales, it becomes increasingly slower for larger scales.

There is a simple cure for this problem. We need a diffusion process where the diffusion constant increases with time. We first discuss a diffusion coefficient that increases linearly with time. This approach results in the differential equation

$$\frac{\partial g}{\partial t} = D_0 t \Delta g. \tag{5.27}$$

A spatial Fourier transform results in

$$\frac{\partial \hat{g}(\boldsymbol{k})}{\partial t} = -4\pi^2 D_0 t |\boldsymbol{k}|^2 \hat{g}(\boldsymbol{k}). \tag{5.28}$$

This equation has the general solution

$$\hat{g}(\boldsymbol{k}, t) = \exp(-2\pi^2 D_0 t^2 |\boldsymbol{k}|^2) \hat{g}(\boldsymbol{k}, 0) \tag{5.29}$$

which is equivalent to a convolution in the spatial domain. Thus,

$$g(\boldsymbol{x}, t) = \frac{1}{2\pi D_0 t^2} \exp\left(-\frac{|\boldsymbol{x}|^2}{2D_0 t^2}\right) * g(\boldsymbol{x}, 0). \tag{5.30}$$

From these equations we can write the convolution kernel and transfer function in the same form as in Eqs. (5.20) and (5.21) with the only exception that the scale parameter

$$\xi_q = D_0 t^2. \tag{5.31}$$

Now the standard deviation for the smoothing is proportional to time for a diffusion process that increases linearly in time. As the scale parameter ξ is proportional to the time squared, we denote this scale space as the *quadratic scale space*. This modified scale space still meets the minimum-maximum principle and the semi-group property.

For even more accelerated smoothing, we can construct a *logarithmic scale space*, i. e., a scale space where the logarithm of the scale parameter increases linearly with time. We use a diffusion coefficient that increases exponentially in time

$$\frac{\partial g}{\partial t} = D_0 \exp(t/\tau)\Delta g. \tag{5.32}$$

Again, we obtain a convolution kernel and a transfer function as in Eqs. (5.20) and (5.21), now with the scale parameter

$$\xi_l = 2D_0\tau \exp(t/\tau). \tag{5.33}$$

5.2.4 Differential Scale Spaces‡

The interest in a *differential scale space* stems from the fact that we want to select optimum scales for processing of features in images. In a differential scale space, the change of the image with scale is emphasized. We use the transfer function of the scale space kernel Eq. (5.21) which is also valid for quadratic and logarithmic scale spaces. The general solution for the scale space can be written in the Fourier space as

$$\hat{g}(\boldsymbol{k}, \xi) = \exp(-2\pi^2|\boldsymbol{k}|^2\xi)\hat{g}(\boldsymbol{k}, 0). \tag{5.34}$$

Differentiating this signal with respect to the scale parameter ξ yields

$$\frac{\partial \hat{g}(\boldsymbol{k}, \xi)}{\partial \xi} = -2\pi^2|\boldsymbol{k}|^2 \exp(-2\pi^2|\boldsymbol{k}|^2\xi)\hat{g}(\boldsymbol{k}, 0) = -2\pi^2|\boldsymbol{k}|^2\hat{g}(\boldsymbol{k}, \xi). \tag{5.35}$$

The multiplication with $-|\boldsymbol{k}|^2$ is equivalent to a second-order spatial derivative (≻ R4), the *Laplacian operator*. Thus we can write in the spatial domain

$$\frac{\partial g(\boldsymbol{x}, \xi)}{\partial \xi} = \frac{1}{2}\Delta g(\boldsymbol{x}, \xi). \tag{5.36}$$

Equations (5.35) and (5.36) constitute a basic property of the differential scale space. The differential scale space is equivalent to a second-order spatial derivation with the Laplacian operator and thus leads to an isotropic *bandpass decomposition* of the image. The transfer function at the scale ξ is

$$-2\pi^2|\boldsymbol{k}|^2 \exp(-2\pi^2|\boldsymbol{k}|^2\xi). \tag{5.37}$$

For small wave numbers, the transfer function is proportional to $-|\boldsymbol{k}|^2$. It reaches a maximum at

$$k^2_{\text{max}} = \frac{2}{\xi} \tag{5.38}$$

and then decays exponentially.

5.2.5 Discrete Scale Spaces‡

The construction of a *discrete scale space* requires a discretization of the diffusion equation. We start with a discretization of the one-dimensional diffusion equation

$$\frac{\partial g(x,\xi)}{\partial \xi} = D\frac{\partial^2 g(x,\xi)}{\partial x^2}. \tag{5.39}$$

The derivatives are replaced by discrete differences in the following way:

$$\begin{aligned} \frac{\partial g(x,\xi)}{\partial \xi} &= \frac{g(x,\xi+\Delta\xi)-g(x,\xi)}{\Delta\xi} \\ \frac{\partial^2 g(x,\xi)}{\partial x^2} &= \frac{g(x+\Delta x,\xi)-2g(x,\xi)+g(x-\Delta x,\xi)}{\Delta x^2}. \end{aligned} \tag{5.40}$$

This leads to the following iteration scheme for computing a discrete scale space with $\epsilon = D\Delta\xi/\Delta x^2$:

$$g(x,\xi+\Delta\xi) = \epsilon g(x+\Delta x,\xi) + (1-2\epsilon)g(x,\xi) + \epsilon g(x-\Delta x,\xi) \tag{5.41}$$

or written with discrete coordinates

$${}^{i+1}g_n = \epsilon\, {}^{i}g_{n+1} + (1-2\epsilon)\, {}^{i}g_n + \epsilon\, {}^{i}g_{n-1}. \tag{5.42}$$

Lindeberg [113] shows that this iteration results in a discrete scale space that meets the minimum-maximum principle and the semi-group property if and only if

$$\epsilon \le 1/4. \tag{5.43}$$

The limiting case of $\Delta\xi = 1/4$ leads to the especially simple iteration

$${}^{i+1}g_n = 1/4\, {}^{i}g_{n+1} + 1/2\, {}^{i}g_n + 1/4\, {}^{i}g_{n-1}. \tag{5.44}$$

Each step of the scale space computation is given by a spatial smoothing of the signal with the mask $\boldsymbol{B}^2 = [1/4\ 1/2\ 1/4]$. We can also formulate the general scale space generating operator in Eq. (5.41) using the convolution operator $\mathcal{B}$. Written in the operator notation introduced in Section 4.1.4, the operator for one iteration step to generate the discrete scale space is

$$(1-4\epsilon)\mathcal{I} + 4\epsilon\mathcal{B}^2 \quad \text{with} \quad \epsilon \le 1/4, \tag{5.45}$$

where $\mathcal{I}$ denotes the identity operator.

This expression is significant, as it can be extended directly to higher dimensions by replacing $\mathcal{B}^2$ with a correspondingly higher-dimensional smoothing operator. The convolution mask $\boldsymbol{B}^2$ is the simplest mask in the class of smoothing binomial filters. These filters will be discussed in detail in Section 11.4.

5.3 Multigrid Representations

5.3.1 Introduction

The scale spaces discussed so far have one significant disadvantage. The use of the additional scale parameter adds a new dimension to the images and thus leads to an explosion of the data storage requirements

and in turn the computational overhead for generating the scale space and for analyzing it. This difficulty is the starting point for a new data structure for representing image data at different scales, known as a *multigrid representation*.

The basic idea is quite simple. While the representation of fine scales requires the full resolution, coarser scales can be represented at lower resolution. This leads to a scale space with smaller and smaller images as the scale parameter increases. In the following two sections we will discuss the *Gaussian pyramid* (Section 5.3.2) and the *Laplacian pyramid* (Section 5.3.3) as efficient discrete implementations of discrete scale spaces. While the Gaussian pyramid constitutes a standard scale space (Section 5.2.1), the Laplacian pyramid is a discrete version of a differential scale space (Section 5.2.4). In this section, we only discuss the basics of multigrid representations. Optimal multigrid smoothing filters are elaborated in Section 11.6.

These pyramids are examples of multigrid data structures that have been introduced into digital image processing in the early 1980s and have led to a tremendous increase in speed of image processing algorithms in digital image processing since then.

5.3.2 Gaussian Pyramid

If we want to reduce the size of an image, we cannot just *subsample* the image by taking, for example, every second pixel in every second line. If we did so, we would disregard the *sampling theorem* (Section 9.2.3). For example, a structure which is sampled three times per wavelength in the original image would only be sampled one and a half times in the subsampled image and thus appear as an aliased pattern as we will discuss in Section 9.1. Consequently, we must ensure that all structures which are sampled less than four times per wavelength are suppressed by an appropriate smoothing filter to ensure a proper subsampled image. For the generation of the scale space, this means that size reduction must go hand in hand with appropriate smoothing.

Generally, the requirement for the smoothing filter can be formulated as

$$\hat{B}(\tilde{\boldsymbol{k}}) = 0 \quad \forall \tilde{k}_p \geq \frac{1}{r_p}, \tag{5.46}$$

where r_p is the subsampling rate in the direction of the pth coordinate.

The combined smoothing and size reduction can be expressed in a single operator by using the following notation to compute the $q+1$th level of the Gaussian pyramid from the qth level:

$$\boldsymbol{G}^{(q+1)} = \mathcal{B}_{\downarrow 2}\boldsymbol{G}^{(q)}. \tag{5.47}$$

The number behind the $\downarrow$ in the index denotes the subsampling rate. The 0th level of the pyramid is the original image: $\boldsymbol{G}^{(0)} = \boldsymbol{G}$.

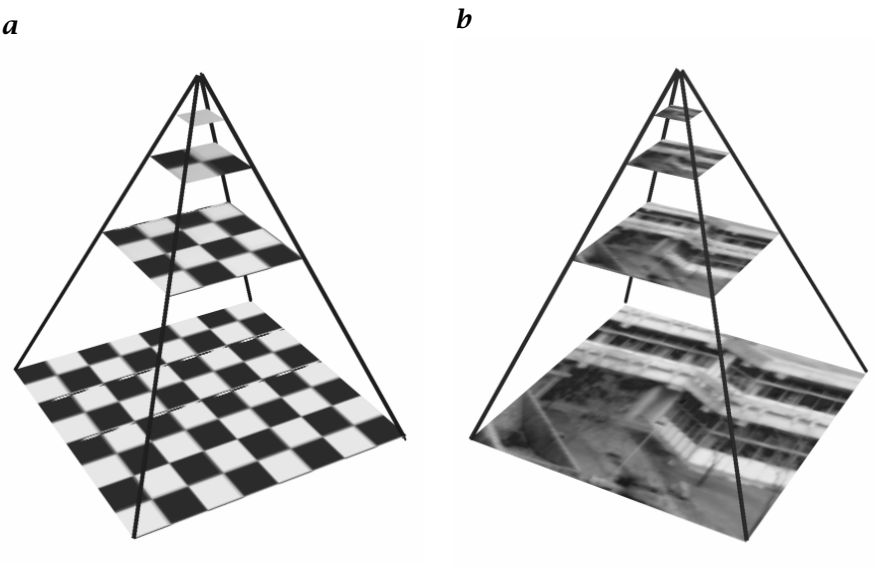

Figure 5.5: *Gaussian pyramid:* ***a*** *schematic representation, the squares of the checkerboard corresponding to pixels;* ***b*** *example.*

If we repeat the smoothing and subsampling operations iteratively, we obtain a series of images, which is called the *Gaussian pyramid*. From level to level, the resolution decreases by a factor of two; the size of the images decreases correspondingly. Consequently, we can think of the series of images as being arranged in the form of a pyramid as illustrated in Fig. 5.5.

The pyramid does not require much storage space. Generally, if we consider the formation of a pyramid from a W-dimensional image with a subsampling factor of two and M pixels in each coordinate direction, the total number of pixels is given by

$$M^W \left(1 + \frac{1}{2^W} + \frac{1}{2^{2W}} + \ldots\right) < M^W \frac{2^W}{2^W - 1}. \tag{5.48}$$

For a two-dimensional image, the whole pyramid needs only $1/3$ more space than the original image for a three-dimensional image only $1/7$ more. Likewise, the computation of the pyramid is equally effective. The *same* smoothing filter is applied to each level of the pyramid. Thus the computation of the *whole* pyramid only needs $4/3$ and $8/7$ times more operations than for the first level of a two-dimensional and three-dimensional image, respectively.

The pyramid brings large scales into the range of local neighborhood operations with small kernels. Moreover, these operations are performed efficiently. Once the pyramid has been computed, we can per-

form neighborhood operations on large scales in the upper levels of the pyramid — because of the smaller image sizes — much more efficiently than for finer scales.

The Gaussian pyramid constitutes a series of lowpass-filtered images in which the cut-off wave numbers decrease by a factor of two (an octave) from level to level. Thus only the coarser details remain in the smaller images (Fig. 5.5). Only a few levels of the pyramid are necessary to span a wide range of wave numbers. From a 512×512 image we can usefully compute only a seven-level pyramid. The smallest image is then 8×8.

5.3.3 Laplacian Pyramid

From the Gaussian pyramid, another pyramid type can be derived, the *Laplacian pyramid*. This type of pyramid is the discrete counterpart to the *differential scale space* discussed in Section 5.2.4 and leads to a sequence of bandpass-filtered images. In contrast to the Fourier transform, the Laplacian pyramid only leads to a coarse wave number decomposition without a directional decomposition. All wave numbers, independently of their direction, within the range of about an octave (factor of two) are contained in one level of the pyramid.

Because of the coarse wave number resolution, we can preserve a good spatial resolution. Each level of the pyramid only contains matching scales which are sampled a few times (two to six) per wavelength. In this way, the Laplacian pyramid is an efficient data structure well adapted to the limits of the product of wave number and spatial resolution set by the *uncertainty relation* (Theorem 7, p. 55).

The differentiation in scale direction in the continuous scale space is approximated by subtracting two levels of the Gaussian pyramid. In order to do so, first the image at the coarser level must be expanded. This operation is performed by an *expansion operator* $\uparrow_2$. As with the reducing smoothing operator, the degree of expansion is denoted by the figure after the $\uparrow$ in the index.

The expansion is significantly more difficult than the size reduction as the missing information must be interpolated. For a size increase of two in all directions, first every second pixel in each row must be interpolated and then every second row. Interpolation is discussed in detail in Section 10.6. With the introduced notation, the generation of the pth level of the Laplacian pyramid can be written as:

$$\boldsymbol{L}^{(p)} = \boldsymbol{G}^{(p)} - \uparrow_2 \boldsymbol{G}^{(p+1)}. \tag{5.49}$$

The Laplacian pyramid is an effective scheme for a *bandpass decomposition* of an image. The center wave number is halved from level to level. The last image of the Laplacian pyramid is a lowpass-filtered image containing only the coarsest structures.

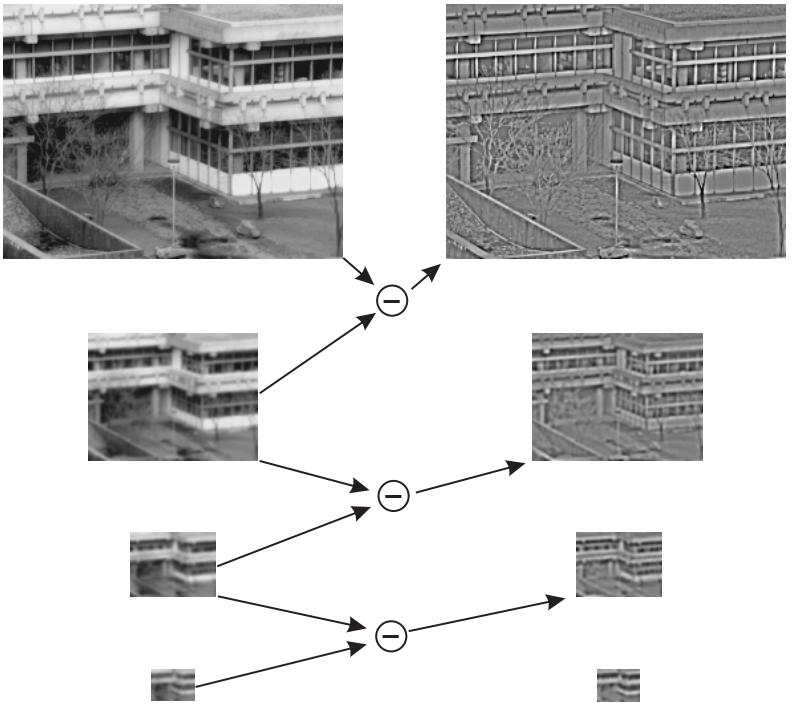

Figure 5.6: *Construction of the Laplacian pyramid (right column) from the Gaussian pyramid (left column) by subtracting two consecutive planes of the Gaussian pyramid.*

The Laplacian pyramid has the significant advantage that the original image can be reconstructed quickly from the sequence of images in the Laplacian pyramid by recursively expanding the images and summing them up. The recursion is the inverse of the recursion in Eq. (5.49). In a Laplacian pyramid with $p+1$ levels, the level p (counting starts with zero!) is the coarsest level of the Gaussian pyramid. Then the level $p-1$ of the Gaussian pyramid can be reconstructed by

$$\boldsymbol{G}^{(p-1)} = \boldsymbol{L}^{(p-1)} + \uparrow_2 \boldsymbol{G}^p \tag{5.50}$$

Note that this is just an inversion of the construction scheme for the Laplacian pyramid. This means that even if the interpolation algorithms required to expand the image contain errors, they affect only the Laplacian pyramid and not the reconstruction of the Gaussian pyramid from the Laplacian pyramid, as the same algorithm is used. The recursion in Eq. (5.50) is repeated with lower levels until level 0, i. e., the original image, is reached again. As illustrated in Fig. 5.6, finer and finer details become visible during the reconstruction process. Because of the pro-

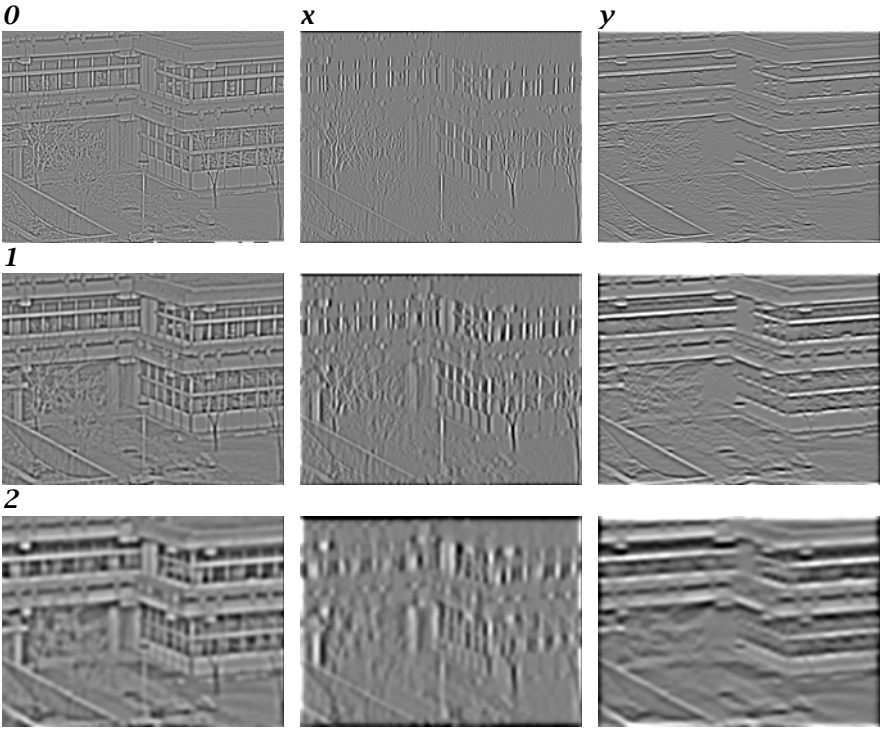

Figure 5.7: *First three planes of a directiopyramidal decomposition of Fig. 5.3a: the rows show are planes 0, 1, and 2, the columns* $\mathbf{L}$, $\mathbf{L}_x$, $\mathbf{L}_y$ *according to Eqs. (5.52) and (5.53).*

gressive reconstruction of details, the Laplacian pyramid has been used as a compact scheme for image compression. Nowadays, more efficient schemes are available on the basis of wavelet transforms, but they operate on principles very similar to those of the Laplacian pyramid.

5.3.4 Directio-Pyramidal Decomposition

In multidimensional signals a directional decomposition is equally important as a scale decomposition. Directional decompositions require suitable directional filters. Ideally, all directional components should add up to the complete image. A combined decomposition of an image into a pyramid and on each pyramid level into directional components is known as a *directiopyramidal decomposition* [78]. Generally, such a decomposition is a difficult filter design problem. Therefore, we illustrate a directiopyramidal decomposition here only with a simple and efficient decomposition scheme with two directional components.

The smoothing is performed by separable smoothing filters, one filter that smoothes only in the x direction ($\mathcal{B}_x$) and one that smoothes only in the y direction ($\mathcal{B}_y$): then the next higher level of the Gaussian pyramid is given as in Eq. (5.47) by

$$\boldsymbol{G}^{(q+1)} = \downarrow_2 \mathcal{B}_x \mathcal{B}_y \boldsymbol{G}^{(q)}. \tag{5.51}$$

The Laplacian pyramid is

$$\boldsymbol{L}^{(q)} = \boldsymbol{G}^{(q)} - \uparrow_2 \boldsymbol{G}^{(q+1)}. \tag{5.52}$$

Then, the two directional components are given by

$$\begin{aligned} \boldsymbol{L}_x^{(q)} &= 1/2(\boldsymbol{G}^{(q)} - \uparrow_2 \boldsymbol{G}^{(q+1)} - (\mathcal{B}_x - \mathcal{B}_y)\boldsymbol{G}^{(q)}), \\ \boldsymbol{L}_y^{(q)} &= 1/2(\boldsymbol{G}^{(q)} - \uparrow_2 \boldsymbol{G}^{(q+1)} + (\mathcal{B}_x - \mathcal{B}_y)\boldsymbol{G}^{(q)}). \end{aligned} \tag{5.53}$$

From Eq. (5.53) it is evident that the two directional components $\boldsymbol{L}_x$ and $\boldsymbol{L}_y$ add up to the isotropic Laplacian pyramid: $\boldsymbol{L} = \boldsymbol{L}_x + \boldsymbol{L}_y$. Example images with the first three levels of a directional decomposition are shown in Fig. 5.7.

5.4 Further Readings‡

Multiresolutional image processing developed in the early 1980ies. An excellent overview of this early work is given by Rosenfeld [156]. Linear scale spaces are described in detail by the monograph of Lindeberg [113], nonlinear scale spaces including inhomogeneous and anisotropic diffusion by Weickert [196]. Readers interested in the actual development of scale space theory are referred to the proceedings of the international conferences on "Scale-Space": 1997 [180], 1999 [131], and 2001 [95].

Part II

Image Formation and Preprocessing

6 Quantitative Visualization

6.1 Introduction

An imaging system collects radiation emitted by objects to make them visible. The radiation consists of a flow of particles or electromagnetic or acoustic waves. In classical computer vision scenes and illumination are taken and analyzed as they are given, but visual systems used in scientific and industrial applications require a different approach. There, the first task is to establish the quantitative relation between the object feature of interest and the emitted radiation. It is the aim of these efforts to map the object feature of interest with minimum possible distortion of the collected radiance by other parameters.

Figure 6.1 illustrates that both the incident ray and the ray emitted by the object towards the camera may be influenced by additional processes. The position of the object can be shifted by refraction of the emitted ray. Scattering and absorption of the incident and emitted rays lead to an attenuation of the radiant flux that is not caused by the observed object itself but by the environment, which thus falsifies the observation. In a proper setup it is important to ensure that these additional influences are minimized and that the received radiation is directly related to the object feature of interest. In cases where we do not have any influence on the illumination or setup, we can still choose radiation of the most appropriate type and wavelength range.

As illustrated in Sections 1.2 and 6.4, a wealth of phenomena is available for imaging objects and object features, including self-emission, induced emission (fluorescence), reflection, refraction, absorption, and scattering of radiation. These effects depend on the optical properties of the object material and on the surface structure of the object. Basically, we can distinguish between surface-related effects caused by discontinuity of optical properties at the surface of objects and volume-related effects.

It is obvious that the complexity of the procedures for quantitative visualization strongly depends on the image processing task. If our goal is only to make a precise geometrical measurement of the objects, it is sufficient to set up an illumination in which the objects are uniformly illuminated and clearly distinguished from the background. In this case, it is not required that we establish quantitative relations between the object features of interest and the radiation emitted towards the camera.

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2

Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

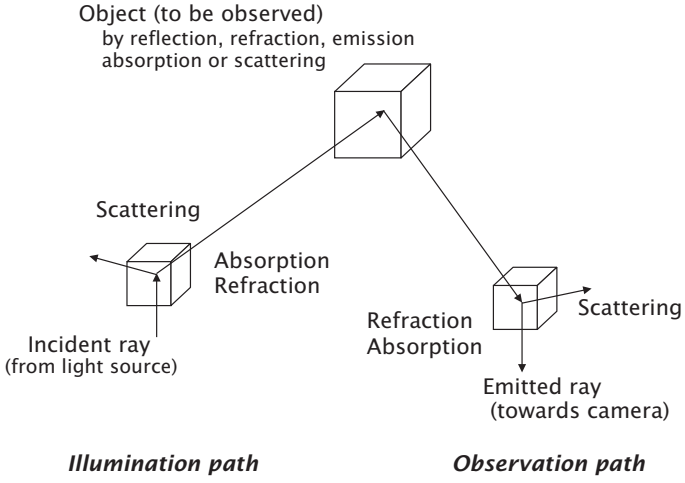

Figure 6.1: *Schematic illustration of the interaction between radiation and matter for the purpose of object visualization. The relation between the emitted radiation towards the camera and the object feature can be disturbed by scattering, absorption, and refraction of the incident and the emitted ray.*

If we want to measure certain object features, however, such as density, temperature, orientation of the surface, or the concentration of a chemical species, we need to know the exact relation between the selected feature and the emitted radiation. A simple example is the detection of an object by its color, i. e., the spectral dependency of the reflection coefficient.

In most applications, however, the relationship between the parameters of interest and the emitted radiation is much less evident. In satellite images, for example, it is easy to recognize urban areas, forests, rivers, lakes, and agricultural regions. But by which features do we recognize them? And, an even more important question, why do they appear the way they do in the images?

Likewise, in medical research one very general question of image-based diagnosis is to detect pathological aberrations. A reliable decision requires a good understanding of the relation between the biological parameters that define the pathological aberration and their appearance in the images.

In summary, essentially two questions must be answered for a successful setup of an imaging system:

1. How does the object radiance (emitted radiative energy flux per solid angle) depend on the object parameters of interest and illumination conditions?

2. How does the irradiance at the image plane (radiative energy flux density) captured by the optical system depend on the object radiance?

This chapter deals with the first of these questions, the second question is addressed in Section 7.5.

6.2 Waves and Particles

Three principal types of radiation can be distinguished: electromagnetic radiation, particulate radiation with atomic or subatomic particles, and acoustic waves. Although these three forms of radiation appear at first glance quite different, they have many properties in common with respect to imaging. First, objects can be imaged by any type of radiation emitted by them and collected by a suitable imaging system.

Second, all three forms of radiation show a wave-like character including particulate radiation. The *wavelength* λ is the distance of one cycle of the oscillation in the propagation direction. The wavelength also governs the ultimate resolution of an imaging system. As a rule of thumb only structures larger than the wavelength of the radiation can be resolved.

Given the different types of radiation, it is obvious that quite different properties of objects can be imaged. For a proper setup of an image system, it is therefore necessary to know some basic properties of the different forms of radiation. This is the purpose of this section.

6.2.1 Electromagnetic Waves

Electromagnetic radiation consists of alternating *electric* and *magnetic fields*. In an *electromagnetic wave*, these fields are directed perpendicular to each other and to the direction of propagation. They are classified by the *frequency* ν and *wavelength* λ. In free space, all electromagnetic waves travel with the *speed of light*, $c \approx 3 \times 10^8$ ms^{-1}. The propagation speed establishes the relation between wavelength λ and frequency ν of an electromagnetic wave as

$$\lambda\nu = c. \tag{6.1}$$

The frequency is measured in cycles per second (Hz or s^{-1}) and the wavelength in meters (m).

As illustrated in Fig. 6.2, electromagnetic waves span an enormous frequency and wavelength range of 24 decades. Only a tiny fraction from about 400–700 nm, about one octave, falls in the visible region, the part to which the human eye is sensitive. The classification usually used for electromagnetic waves (Fig. 6.2) is somewhat artificial and has mainly historical reasons given by the way these waves are generated or detected.

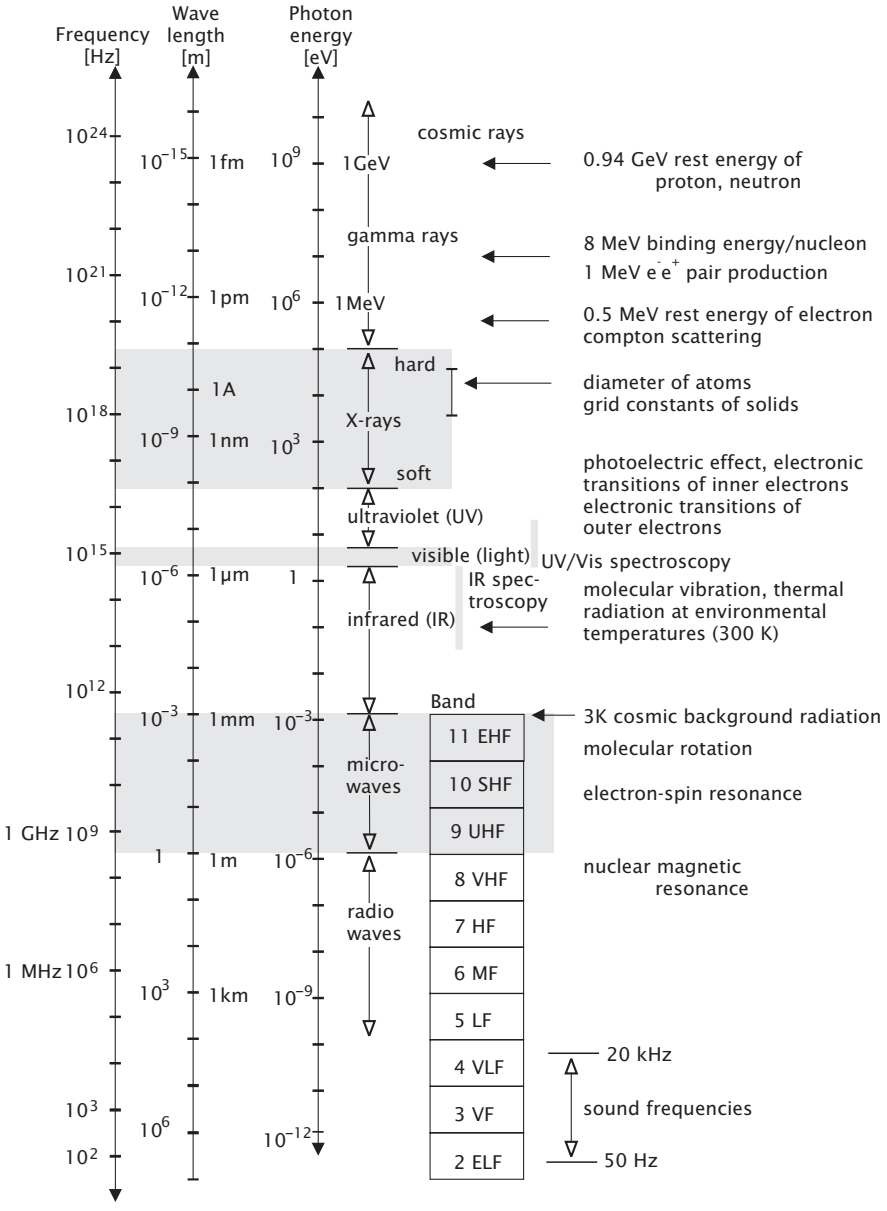

Figure 6.2: *Classification of the electromagnetic spectrum with wavelength, frequency, and photon energy scales.*

In matter, the electric and magnetic fields of the electromagnetic wave interact with the electric charges, electric currents, electric fields, and magnetic fields in the medium. Nonetheless, the basic nature of electromagnetic waves remains the same, only the propagation of the wave is slowed down and the wave is attenuated.

The simplest case is given when the medium reacts in a linear way to the disturbance of the electric and magnetic fields caused by the electromagnetic wave and when the medium is isotropic. Then the influence of the medium is expressed in the complex *index of refraction*, $\eta = n + i\chi$. The real part, n, or ordinary index of refraction, is the ration of the speed of light, c, to the propagation velocity u in the medium, $n = c/u$. The imaginary component of η, χ, is related to the attenuation of the wave amplitude.

Generally, the index of refraction depends on the frequency or wavelength of the electromagnetic wave. Therefore, the propagation speed of a wave is no longer independent of the wavelength. This effect is called *dispersion* and the wave is called a dispersive wave.

The index of refraction and the attenuation coefficient are the two primary parameters characterizing the optical properties of a medium. In the context of imaging they can be used to identify a chemical species or any other physical parameter influencing it.

Electromagnetic waves are generally a linear phenomenon. This means that we can decompose any complex wave pattern into basic ones such as plane harmonic waves. Or, conversely, we can superimpose any two or more electromagnetic waves and be sure that they are still electromagnetic waves.

This superposition principle only breaks down for waves with very high field strengths. Then, the material no longer acts in a linear way on the electromagnetic wave but gives rise to *nonlinear optical phenomena*. These phenomena have become obvious only quite recently with the availability of very intense light sources such as lasers. A prominent nonlinear phenomenon is the *frequency doubling* of light. This effect is now widely used in lasers to produce output beams of double the frequency (half the wavelength). From the perspective of quantitative visualization, these nonlinear effects open an exciting new world for visualizing specific phenomena and material properties.

6.2.2 Polarization

The superposition principle can be used to explain the polarization of electromagnetic waves. Polarization is defined by the orientation of the electric field vector $\boldsymbol{E}$. If this vector is confined to a plane, as in the previous examples of a plane harmonic wave, the radiation is called *plane polarized* or *linearly polarized*. In general, electromagnetic waves are not polarized. To discuss the general case, we consider two waves traveling

in the z direction, one with the electric field component in the x direction and the other with the electric field component in the y direction. The amplitudes E_1 and E_2 are constant and ϕ is the phase difference between the two waves. If $\phi = 0$, the electromagnetic field vector is confined to a plane. The angle ϕ of this plane with respect to the x axis is given by

$$\phi = \arctan \frac{E_2}{E_1}. \tag{6.2}$$

Another special case arises if the phase difference $\phi = \pm 90^\circ$ and $E_1 = E_2$; then the wave is called *circularly polarized*. In this case, the electric field vector rotates around the propagation direction with one turn per period of the wave. The general case where both the phase difference is not $\pm 90^\circ$ and the amplitudes of both components are not equal is called *elliptically polarized*. In this case, the E vector rotates in an ellipse, i. e., with changing amplitude around the propagation direction. It is important to note that any type of polarization can also be composed of a right and a left circular polarized beam. A left circular and a right circular beam of the same amplitude, for instance, combine to form a linear polarized beam. The direction of the polarization plane depends on the phase shift between the two circularly polarized beams.

6.2.3 Coherence

An important property of some electromagnetic waves is their *coherence*. Two beams of radiation are said to be *coherent* if a systematic relationship between the phases of the electromagnetic field vectors exists. If this relationship is random, the radiation is *incoherent*. It is obvious that incoherent radiation superposes in a different way than coherent radiation. In case of coherent radiation, destructive inference is possible in the sense that waves quench each other in certain places were the phase shift is 180°.

Normal light sources are incoherent. They do not send out one continuous planar wave but rather wave packages of short duration and with no particular phase relationship. In contrast, a laser is a coherent light source.

6.2.4 Photons

Electromagnetic radiation has particle-like properties in addition to those characterized by wave motion. Electromagnetic energy is quantized in that for a given frequency its energy can only occur in multiples of the quantity $h\nu$ in which h is *Planck's constant*, the *action quantum*:

$$E = h\nu. \tag{6.3}$$

The quantum of electromagnetic energy is called the *photon*.

In any interaction of radiation with matter, be it absorption of radiation or emission of radiation, energy can only be exchanged in multiples of these quanta. The energy of the photon is often given in the energy unit electron volts (eV), which is the kinetic energy an electron would acquire in being accelerated through a potential difference of one volt. A photon of yellow light, for example, has an energy of approximately 2 eV. Figure 6.2 includes a photon energy scale in eV. The higher the frequency of electromagnetic radiation, the more its particulate nature becomes apparent, because its energy quanta get larger. The energy of a photon can be larger than the energy associated with the rest mass of an elementary particle. In this case it is possible for electromagnetic energy to be spontaneously converted into mass in the form of a pair of particles. Although a photon has no rest mass, a momentum is associated with it, since it moves with the speed of light and has a finite energy. The momentum, p, is given by

$$p = h/\lambda. \tag{6.4}$$

The quantization of the energy of electromagnetic waves is important for imaging since sensitive radiation detectors can measure the absorption of a *single* photon. Such detectors are called photon counters. Thus, the lowest energy amount that can be detected is $h\nu$. The random nature of arrival of photons at the detector gives rise to an uncertainty ("noise") in the measurement of radiation energy. The number of photons counted per unit time is a *random variable* with a *Poisson distribution* as discussed in Section 3.4.1. If N is the average number of counted photons in a given time interval, the Poisson distribution has a standard deviation $\sigma_N = \sqrt{N}$. The measurement of a radiative flux with a relative standard deviation of 1 % thus requires the counting of 10 000 photons.

6.2.5 Particle Radiation

Unlike electromagnetic waves, most *particulate radiation* moves at a speed less than the speed of light because the particles have a non-zero rest mass. With respect to imaging, the most important type of particulate radiation is due to *electrons*, also known as *beta radiation* when emitted by radioactive elements. Other types of important particulate radiation are due to the positively charged nucleus of the hydrogen atom or the *proton*, the nucleus of the helium atom or *alpha radiation* which has a double positive charge, and the *neutron*.

Particulate radiation also shows a wave-like character. The wavelength λ and the frequency ω are directly related to the energy and momentum of the particle:

$$\begin{aligned} \nu &= E/h \quad \text{Bohr frequency condition,} \\ \lambda &= h/p \quad \text{de Broglie wavelength relation.} \end{aligned} \tag{6.5}$$

These are the same relations as for the photon, Eqs. (6.3) and (6.4). Their significance for imaging purposes lies in the fact that particles typically have much shorter wavelength radiation. Electrons, for instance, with an energy of 20 keV have a wavelength of about 10^{-11} m or 10 pm, less than the diameter of atoms (Fig. 6.2) and about 50 000 times less than the wavelength of light. As the resolving power of any imaging system — with the exception of nearfield systems — is limited to scales in the order of a wavelength of the radiation, imaging systems based on electrons such as the *electron microscope*, have a much higher potential resolving power than any light microscope.

6.2.6 Acoustic Waves

In contrast to electromagnetic waves, *acoustic* or *elastic waves* need a carrier. Acoustic waves propagate elastic deformations. So-called *longitudinal acoustic waves* are generated by isotropic pressure, causing a uniform compression and thus a deformation in the direction of propagation. The local density ρ, the local pressure p and the local velocity $\boldsymbol{v}$ are governed by the same wave equation

$$\frac{\partial^2 \rho}{\partial t^2} = u^2 \Delta \rho, \quad \frac{\partial^2 p}{\partial t^2} = u^2 \Delta p, \qquad \text{with} \quad u = \frac{1}{\sqrt{\rho_0 \beta_{ad}}}, \tag{6.6}$$

where u is the velocity of sound, ρ_0 is the static density and β_{ad} the *adiabatic compressibility*. The adiabatic compressibility is given as the relative volume change caused by a uniform pressure (force/unit area) under the condition that no heat exchange takes place:

$$\beta_{ad} = -\frac{1}{V}\frac{\mathrm{d}V}{\mathrm{d}P}. \tag{6.7}$$

Thus the *speed of sound* is related in a universal way to the elastic properties of the medium. The lower the density and the compressibility, the higher is the speed of sound. Acoustic waves travel much slower than electromagnetic waves. Their speed in air, water, and iron at 20°C is 344 m/s, 1485 m/s, and 5100 m/s, respectively. An audible acoustic wave with a frequency of 3 kHz has a wavelength in air of about 10 cm. However, acoustic waves with a much higher frequency, known as *ultrasound*, can have wavelengths down in the micrometer range. Using suitable acoustic lenses, *ultrasonic microscopy* is possible.

If sound or ultrasound is used for imaging, it is important to point out that propagation of sound is much more complex in solids. First, solids are generally not isotropic, and the elasticity of a solid can not be described by a scalar compressibility. Instead, a tensor is required to describe the elasticity properties. Second, shear forces in contrast to pressure forces give rise also to *transversal acoustic waves*, where

the deformation is perpendicular to the direction of propagation as with electromagnetic waves. Thus, sound waves of different modes travel with different velocities in a solid.

Despite all these complexities, the velocity of sound depends only on the density and the elastic properties of the medium. Therefore, acoustic waves show no dispersion, i. e., waves of different frequencies travel with the same speed. This is an important basic fact for *acoustic imaging* techniques.

6.3 Radiometry, Photometry, Spectroscopy, and Color

6.3.1 Radiometry Terms

Radiometry is the topic in optics describing and measuring radiation and its interaction with matter. Because of the dual nature of radiation, the radiometric terms refer either to energy or to particles; in case of electromagnetic radiation, the particles are photons (Section 6.2.4). If it is required to distinguish between the two types, the indices e and p are used for radiometric terms.

Radiometry is not a complex subject. It has only become a confusing subject following different, inaccurate, and often even wrong usage of its terms. Moreover, radiometry is taught less frequently and less thoroughly than other subjects in optics. Thus, knowledge about radiometry is less widespread. However, it is a very important subject for imaging. Geometrical optics only tells us where the image of an object is located, whereas radiometry says how much radiant energy has been collected from an object.

Radiant Energy. Since radiation is a form of energy, it can do work. A body absorbing radiation is heated up. Radiation can set free electric charges in a suitable material designed to detect radiation. *Radiant energy* is denoted by Q and given in units of Ws (joule) or number of particles (photons).

Radiant Flux. The power of radiation, i. e., the energy per unit time, is known as *radiant flux* and denoted by Φ:

$$\Phi = \frac{\mathrm{d}Q}{\mathrm{d}t}. \tag{6.8}$$

This term is important to describe the total energy emitted by a light source per unit time. Its unit is joule/s (Js^{-1}), watt (W), or photons per s (s^{-1}).

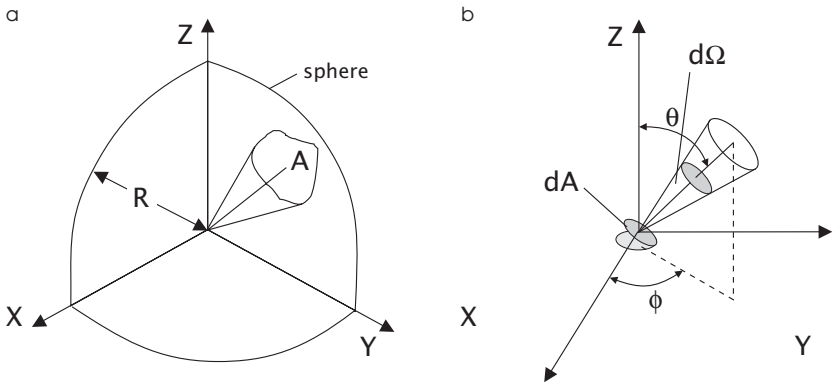

Figure 6.3: ***a*** *Definition of the solid angle.* ***b*** *Definition of radiance, the radiant power emitted per unit surface area* $\mathrm{d}A$ *projected in the direction of propagation per unit solid angle* Ω.

Radiant Flux Density. The radiant flux per unit area, the flux density, is known by two names:

$$\text{irradiance} \quad E = \frac{\mathrm{d}\Phi}{\mathrm{d}A_0}, \quad \text{excitance} \quad M = \frac{\mathrm{d}\Phi}{\mathrm{d}A_0}. \tag{6.9}$$

The *irradiance*, E, is the radiant flux incident upon a surface per unit area, for instance a sensor that converts the radiant energy into an electric signal. The unit of irradiance is $\mathrm{W\,m^{-2}}$, or photons per area and time ($\mathrm{m^{-2}s^{-1}}$). If the radiation is emitted from a surface, the radiant flux density is called *excitance* or *emittance* and denoted by M.

Solid Angle. The concept of the *solid angle* is paramount for an understanding of the angular distribution of radiation. Consider a compact source at the center of a sphere of radius R beaming radiation outwards in a cone of directions (Fig. 6.3a). The boundaries of the cone outline an area A on the sphere. The solid angle (Ω) measured in steradians (sr) is the area A divided by the square of the radius ($\Omega = A/R^2$). Although the steradian is a dimensionless quantity, it is advisable to use it explicitly when a radiometric term referring to a solid angle can be confused with the corresponding non-directional term. The solid angle of a whole sphere and hemisphere are 4π and 2π, respectively.

Radiant Intensity. The (total) radiant flux per unit solid angle emitted by a source is called the *radiant intensity* I:

$$I = \frac{\mathrm{d}\Phi}{\mathrm{d}\Omega}. \tag{6.10}$$

It is obvious that this term only makes sense for describing compact or point sources, i. e., when the distance from the source is much larger

than its size. This region is also often called the far-field of a radiator. Intensity is also useful for describing light beams.

Radiance. For an extended source, the radiation per unit area in the direction of excitance and per unit solid angle is an important quantity (Fig. 6.3b):

$$L = \frac{\mathrm{d}^2\Phi}{\mathrm{d}A\,\mathrm{d}\Omega} = \frac{\mathrm{d}^2\Phi}{\mathrm{d}A_0 \cos\theta\,\mathrm{d}\Omega}. \tag{6.11}$$

The radiation can either be emitted from, pass through, or be incident on the surface. The radiance L depends on the angle of incidence to the surface, θ (Fig. 6.3b), and the azimuth angle ϕ. For a planar surface, θ and ϕ are contained in the interval $[0, \pi/2]$ and $[0, 2\pi]$, respectively. It is important to realize that the radiance is related to a unit area in the direction of excitance, $dA = dA_0 \cdot \cos\theta$. Thus, the effective area from which the radiation is emitted increases with the angle of incidence. The unit for energy-based and photon-based radiance are $\mathrm{W\,m^{-2}sr^{-1}}$ and $\mathrm{s^{-1}m^{-2}sr^{-1}}$, respectively.

Often, radiance — especially incident radiance — is called brightness. It is better not to use this term at all as it has contributed much to the confusion between radiance and irradiance. Although both quantities have the same dimension, they are quite different. Radiance L describes the angular distribution of radiation, while irradiance E integrates the radiance incident to a surface element over a solid angle range corresponding to all directions under which it can receive radiation:

$$E = \int_\Omega L(\theta,\phi)\cos\theta\,\mathrm{d}\Omega = \int_0^{\pi/2}\int_0^{2\pi} L(\theta,\phi)\cos\theta\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi. \tag{6.12}$$

The factor $\cos\theta$ arises from the fact that the unit area for radiance is related to the direction of excitance (Fig. 6.3b), while the irradiance is related to a unit area parallel to the surface.

6.3.2 Spectroradiometry

Because any interaction between matter and radiation depends on the wavelength or frequency of the radiation, it is necessary to treat all radiometric quantities as a function of the *wavelength*. Therefore, we define all these quantities per unit interval of wavelength. Alternatively, it is also possible to use unit intervals of frequencies or wave numbers. The *wave number* denotes the number of wavelengths per unit length interval (see Eq. (2.14) and Section 2.3.4). To keep the various spectral quantities distinct, we specify the dependency explicitly, e. g., $L(\lambda)$, $L(\nu)$, and $L(k)$.

The radiometric terms discussed in the previous section measure the properties of radiation in terms of energy or number of photons. *Photometry* relates the same quantities to the human eyes' response to them. Photometry is of importance to scientific imaging in two respects: First, photometry gives a quantitative approach to radiometric terms as they are sensed by the human eye. Second, photometry serves as a model for how to describe the response of any type of radiation sensor used to convert irradiance into an electric signal. The key in understanding photometry is to look at the spectral response of the human eye. Otherwise, there is nothing new to photometry.

6.3.3 Spectral Sampling Methods

Spectroscopic imaging is in principle a very powerful tool for identifying objects and their properties because almost all optical material constants depend on the wavelength of the radiation. The trouble with spectroscopic imaging is that it adds another coordinate to imaging and the required amount of data is multiplied correspondingly. Therefore, it is important to sample the spectrum with the minimum number of samples sufficient to perform the required task. Here, we introduce several general spectral sampling strategies. In the next section, we also discuss human color vision from this point of view as one realization of spectral sampling.

Line sampling is a technique where each channel picks only a narrow spectral range (Fig. 6.4a). This technique is useful if processes are to be imaged that are related to emission or absorption at specific spectral lines. The technique is very selective. One channel "sees" only a specific wavelength and is insensitive — at least to the degree that such a narrow bandpass filtering can be realized technically — to all other wavelengths. Thus, this technique is suitable for imaging very specific effects or specific chemical species. It cannot be used to make an estimate of the total radiance from objects since it misses most wavelengths.

Band sampling is the appropriate technique if the total radiance in a certain wavelength range has to be imaged and still some wavelength resolution is required (Fig. 6.4b). Ideally, the individual bands have uniform responsivity and are adjacent to each other. Thus, band sampling gives the optimum resolution with a few channels but does not allow any distinction of the wavelengths within one band. The spectral resolution achievable with this sampling method is limited to the width of the spectral bands of the sensors.

In many cases, it is possible to make a model of the spectral radiance of a certain object. Then, a much better spectral sampling technique can be chosen that essentially samples not certain wavelengths but rather the parameters of the model. This technique is known as *model-based spectral sampling*.

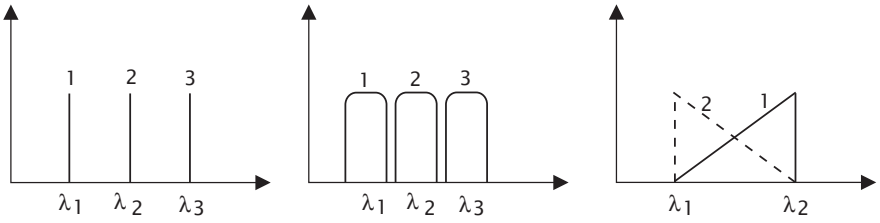

Figure 6.4: *Examples of spectral sampling:* ***a*** *line sampling,* ***b*** *band sampling,* ***c*** *sampling adapted to a certain model of the spectral range, in this example for a single spectral line of unknown wavelength.*

We will illustrate this general approach with a simple example. It illustrates a method for measuring the mean wavelength of an arbitrary spectral distribution $\phi(\lambda)$ and the total radiative flux in a certain wave number range. These quantities are defined as

$$\phi = \frac{1}{\lambda_2 - \lambda_1}\int_{\lambda_1}^{\lambda_2}\phi(\lambda)\,\mathrm{d}\lambda \quad \text{and} \quad \overline{\lambda} = \int_{\lambda_1}^{\lambda_2}\lambda\phi(\lambda)d\lambda \Big/ \int_{\lambda_1}^{\lambda_2}\phi(\lambda)\,\mathrm{d}\lambda\,. \tag{6.13}$$

In the second equation, the spectral distribution is multiplied by the wavelength λ. Therefore, we need a sensor that has a sensitivity varying linearly with the wave number. We try two sensor channels with the following linear spectral *responsivity*, as shown in Fig. 6.4c:

$$\begin{aligned} R_1(\lambda) &= \frac{\lambda-\lambda_1}{\lambda_2-\lambda_1}R_0 = \left(\frac{1}{2}+\tilde{\lambda}\right)R_0 \\ R_2(\lambda) &= R_0 - R_1(\lambda) = \left(\frac{1}{2}-\tilde{\lambda}\right)R_0, \end{aligned} \tag{6.14}$$

where R is the responsivity of the sensor and $\tilde{\lambda}$ the normalized wavelength

$$\tilde{\lambda} = \left(\lambda - \frac{\lambda_1+\lambda_2}{2}\right)/(\lambda_2-\lambda_1). \tag{6.15}$$

$\tilde{\lambda}$ is zero in the middle and $\pm 1/2$ at the edges of the interval.

The sum of the responsivity of the two channels is independent of the wavelength, while the difference is directly proportional to the wavelength and varies from $-R_0$ for $\lambda = \lambda_1$ to R_0 for $\lambda = \lambda_2$:

$$\begin{aligned} R_1'(\tilde{\lambda}) &= R_1(\lambda) + R_2(\lambda) = R_0 \\ R_2'(\tilde{\lambda}) &= R_1(\lambda) - R_2(\lambda) = 2\tilde{\lambda}R_0. \end{aligned} \tag{6.16}$$

Thus the sum of the signals from the two sensors R_1 and R_2, gives the total radiative flux, while the mean wavelength is given by $2\tilde{\lambda} = (R_1 - R_2)/(R_1 + R_2)$. Except for these two quantities, the sensors cannot reveal any further details about the spectral distribution.

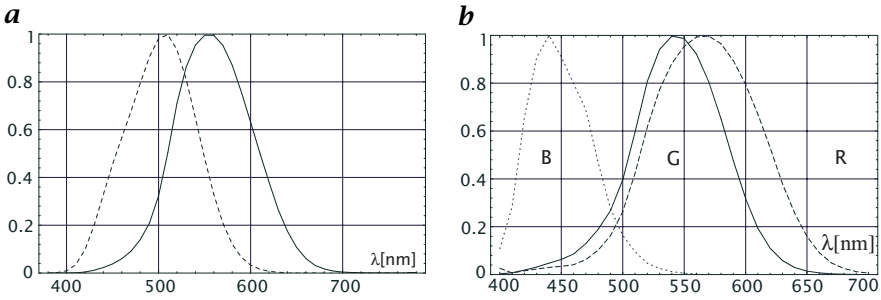

Figure 6.5: ***a*** *Relative spectral response of the "standard" human eye as set by the CIE in 1980 under medium to high irradiance levels (photopic vision, $V(\lambda)$, solid line), and low radiance levels (scotopic vision, $V'(\lambda)$, dashed line); data from [105].* ***b*** *Relative cone sensitivities of the human eye after DeMarco et al. [28].*

6.3.4 Human Color Vision

The *human visual system* responds only to electromagnetic radiation having wavelengths between about 360 and 800 nm. It is very insensitive at wavelengths between 360 and about 410 nm and between 720 and 830 nm. Even for individuals without vision defects, there is some variation in the spectral response. Thus, the visible range in the electromagnetic spectrum (light, Fig. 6.2) is somewhat uncertain.

The retina of the eye onto which the image is projected contains two general classes of receptors, rods and cones. Photopigments in the outer segments of the receptors absorb radiation. The absorbed energy is then converted into neural electrochemical signals which are transmitted via subsequent neurons and the optic nerve to the brain. Three different types of photopigments in the cones make them sensitive to different spectral ranges and, thus, enable color vision (Fig. 6.5b). Vision with cones is only active at high and medium illumination levels and is also called *photopic vision*. At low illumination levels, vision is taken over by the rods. This type of vision is called *scotopic vision*.

At first glance it might seem impossible to measure the spectral response of the eye in a quantitative way since we can only rely on the subjective impression how the human eye senses "radiance". However, the spectral response of the human eye can be measured by making use of the fact that it can sense brightness differences very sensitively. Based on extensive studies with many individuals, in 1924 the International Lighting Commission (CIE) set a standard for the spectral response of the human observer under photopic conditions that was slightly revised several times later on. Figure 6.5 show the 1980 values. The relative spectral response curve for scotopic vision, $V'(\lambda)$ is similar in shape but the peak is shifted from about 555 nm to 510 nm (Fig. 6.5a).

Physiological measurements can only give a relative spectral luminous efficiency function. Therefore, it is required to set a new unit for luminous quantities. This new unit is the candela; it is one of the seven fundamental units of the metric system (Système Internationale, or SI). The candela is defined to be the luminous intensity of a monochromatic source with a frequency of 5.4×10^{14} Hz and a radiant intensity of 1/683 W/sr. The odd factor 1/683 has historical reasons because the candela was previously defined independently from radiant quantities.

With this definition of the luminous intensity and the capability of the eye to detect small changes in brightness, the luminous intensity of any light source can be measured by comparing it to a standard light source. This approach, however, would refer the luminous quantities to an individual observer. Therefore, it is much better to use the standard spectral luminous efficacy function. Then, any luminous quantity can be computed from its corresponding radiometric quantity by:

$$\begin{aligned} Q_v &= 683 \frac{\mathrm{lm}}{\mathrm{W}} \int_{380\,\mathrm{nm}}^{780\,\mathrm{nm}} Q(\lambda) V(\lambda)\, \mathrm{d}\lambda \quad \text{photopic,} \\ Q_{v'} &= 1754 \frac{\mathrm{lm}}{\mathrm{W}} \int_{380\,\mathrm{nm}}^{780\,\mathrm{nm}} Q(\lambda) V'(\lambda)\, \mathrm{d}\lambda \quad \text{scotopic,} \end{aligned} \tag{6.17}$$

where $V(\lambda)$ is the spectral luminous efficacy for day vision (photopic). A list with all photometric quantities and their radiant equivalent can be found in Appendix A (≻ R15). The units of luminous flux, the photometric quantity equivalent to radiant flux (units W) is lumen (lm).

In terms of the spectral sampling techniques summarized above, human color vision can be regarded as a blend of band sampling and model-based sampling. The sensitivities cover different bands with maximal sensitivities at 445 nm, 535 nm, and 575 nm, respectively, but which overlap each other significantly (Fig. 6.5b). In contrast to our model examples, the three sensor channels are unequally spaced and cannot simply be linearly related. Indeed, the color sensitivity of the human eye is uneven, and all the nonlinearities involved make the science of color vision rather difficult. Here, we give only some basic facts in as much as they are useful to handle color images.

With three color sensors, it is obvious that color signals cover a 3-D space. Each point in this space represents one color. It is clear that many spectral distributions, known as *metameric color stimuli* or just metameres, map onto one point in the color space. Generally, we can write the signal s_i received by a sensor with a spectral responsivity $R_i(\lambda)$ as

$$s_i = \int R_i(\lambda) \phi(\lambda)\, \mathrm{d}\lambda. \tag{6.18}$$

With three primary color sensors, a triple of values is received, often called a *tristimulus*.

One of the most important questions in *colorimetry* is how to set up a system representing colors as linear combination of some basic or *primary colors*. A set of three spectral distributions $\phi_j(\lambda)$ represents a set of primary colors and results in an array of responses that can be described by the matrix P with

$$p_{ij} = \int R_i(\lambda)\phi_j(\lambda)\,\mathrm{d}\lambda. \tag{6.19}$$

Each vector $\boldsymbol{p}_j = (p_{1j}, p_{2j}, p_{3j})$ represents a tristimulus of the primary colors in the 3-D color space. It is obvious that only colors can be represented that are a linear combination of the base vectors $\boldsymbol{p}_j$

$$\boldsymbol{s} = R\boldsymbol{p}_1 + G\boldsymbol{p}_2 + B\boldsymbol{p}_3 \qquad \text{with} \qquad 0 \le R, G, B \le 1 \tag{6.20}$$

where the coefficients are denoted by R, G, and B, indicating the three primary colors red, green, and blue. Only if the three base vectors $\boldsymbol{p}_j$ are an orthogonal base can all colors be presented as a linear combination of them. One possible and easily realizable primary color system is formed by the monochromatic colors red, green, and blue with wavelengths 700 nm, 546.1 nm, and 435.8 nm, as adopted by the CIE in 1931. In the following, we use the primary color system according to the European EBU norm with red, green, and blue phosphor, as this is the standard way color images are displayed.

Given the significant overlap in the spectral response of the three types of cones (Fig. 6.5b), especially in the green image, it is obvious that no primary colors exist that can span the color systems. The colors that can be represented lie within the parallelepiped formed by the three base vectors of the primary colors. The more the primary colors are correlated with each other, i. e., the smaller the angle between two of them, the smaller is the color space that can be represented by them. Mathematically, colors that cannot be represented by a set of primary colors have at least *one* negative coefficient in Eq. (6.20).

One component in the 3-D color space is intensity. If a color vector is multiplied by a scalar, only its intensity is changed but not the color. Thus, all colors could be normalized by the intensity. This operation reduces the 3-D color space to a 2-D color plane or chromaticity diagram:

$$r = \frac{R}{R+G+B}, \quad g = \frac{G}{R+G+B}, \quad b = \frac{B}{R+G+B}, \tag{6.21}$$

$$\text{with} \qquad r + g + b = 1. \tag{6.22}$$

It is sufficient to use only the two components r and g. The third component is then given by $b = 1 - r - g$, according to Eq. (6.22). Thus,

all colors that can be represented by the three primary colors R, G, and B are confined within a triangle in the rg space as shown in Fig. 6.6a. As already mentioned, some colors cannot be represented by the primary colors. The boundary of all possible colors is given by the visible monochromatic colors from deep red to blue. The line of monochromatic colors form a U-shaped curve in the rg-space. Since all colors that lie on a straight line between two colors can be generated as an additive mixture of these colors, the space of all possible colors covers the area filled by the U-shaped spectral curve and the straight mixing line between its two end points for blue and red color (purple line).

In order to avoid negative color coordinate values, often a new coordinate system is chosen with virtual primary colors, i. e., primary colors that cannot be realized by any physical colors. This color system is known as the *XYZ color system* and constructed in such a way that it just includes the curve of monochromatic colors with only positive coefficients (Fig. 6.6c) and given by the following linear coordinate transform:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.490 & 0.310 & 0.200 \\ 0.177 & 0.812 & 0.011 \\ 0.000 & 0.010 & 0.990 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{6.23}$$

The back-transform from the XYZ color system to the RGB color system is given by the inverse of the matrix in Eq. (6.23).

The color systems discussed so far do not directly relate to the human sense of color. From the rg or xy values, we cannot directly infer colors such as green or blue. A natural type of description of colors includes besides the *luminance* (*intensity*) the type of color, such as green or blue (*hue*) and the purity of the color (*saturation*). From a pure color, we can obtain any degree of saturation by mixing it with white.

Hue and saturation can be extracted from chromaticity diagrams by simple coordinate transformations. The point of reference is the white point in the middle of the chromaticity diagram (Fig. 6.6b). If we draw a line from this point to a pure (monochromatic) color, it constitutes a mixing line for a pure color with white and is thus a line of constant hue. From the white point to the pure color, the saturation increases linearly. The *white point* is given in the rg chromaticity diagram by $w = [1/3, 1/3]^T$.

A color system that has its center at the white point is called a *color difference system*. From a color difference system, we can infer a hue-saturation color system (hue, saturation, and density; HIS) by simply using polar coordinate systems. Then, the saturation is proportional to the radius and the hue to the angle (Fig. 6.6b).

So far, color science is easy. All the real difficulties arise from the need to adapt the color system in an optimum way to display and print devices and for transmission by television signals or to correct for the

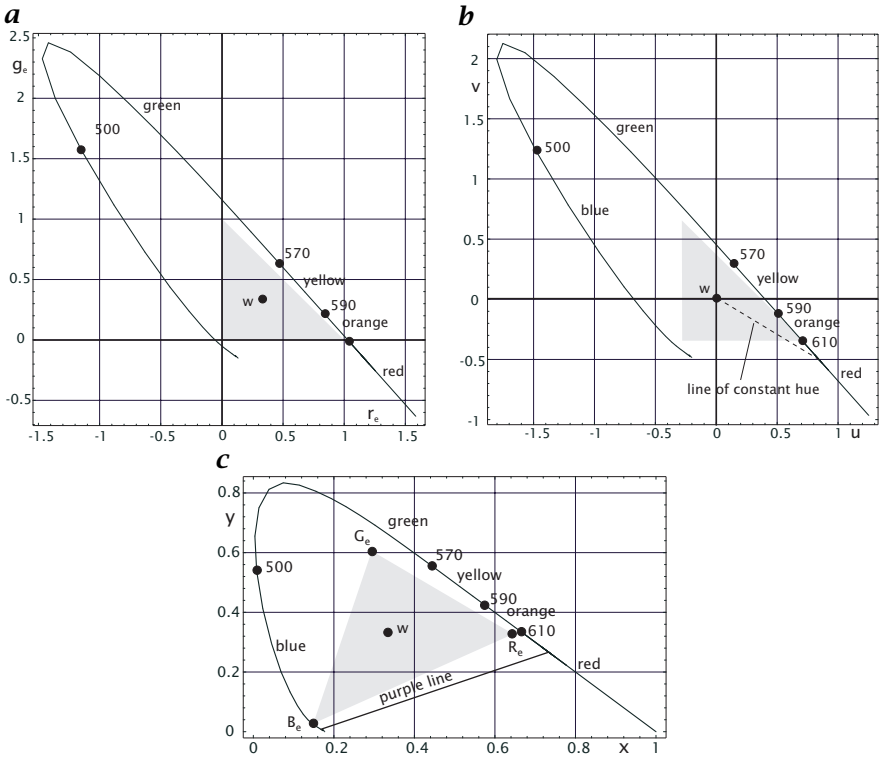

Figure 6.6: *Chromaticity diagram shown in the* ***a*** rg*-color space,* ***b*** uv*-color space,* ***c*** xy*-color space; the shaded triangles indicate the colors that can be generated by additive color mixing using the primary colors R, G, and B.*

uneven color resolution of the human visual system that is apparent in the chromaticity diagrams of simple color spaces (Fig. 6.6). These needs have led to a confusing variety of different color systems (➢ R16).

6.4 Interactions of Radiation with Matter‡

The interaction of radiation with matter is the basis for any imaging technique. Basically, two classes of interactions of radiation with matter can be distinguished. The first class is related to the discontinuities of optical properties at the interface between two different materials (Fig. 6.7a). The second class is volume-related and depends on the optical properties of the material (Fig. 6.7b). In this section, we give a brief summary of the most important phenomena. The idea is to give the reader an overview of the many possible ways to measure material properties with imaging techniques.

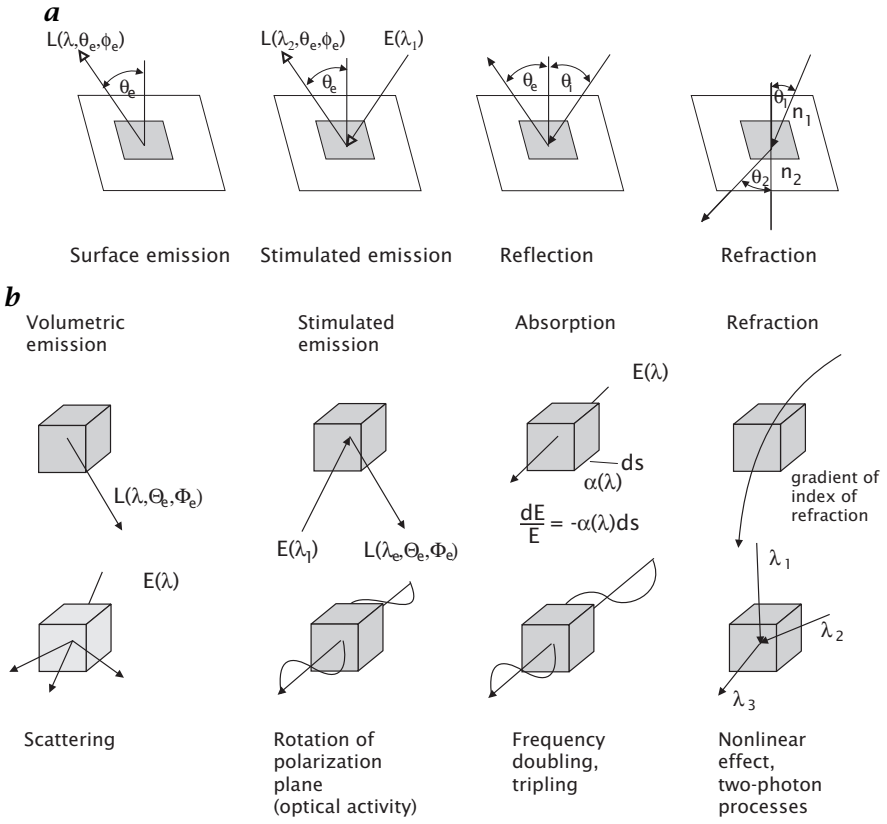

Figure 6.7: *Principle possibilities for interaction of radiation and matter:* ***a*** *at the surface of an object, i. e., at the discontinuity of optical properties,* ***b*** *volume related.*

6.4.1 Thermal Emission‡

Emission of electromagnetic radiation occurs at any temperature and is thus a ubiquitous form of interaction between matter and electromagnetic radiation. The cause for the spontaneous emission of electromagnetic radiation is thermal molecular motion, which increases with temperature. During emission of radiation, thermal energy is converted to electromagnetic radiation and, according to the universal law of energy conservation, the matter is cooling down.

An upper level for thermal emission exists. According to the laws of thermodynamics, the fraction of radiation at a certain wavelength that is absorbed must also be re-emitted: thus, there is an upper limit for the emission, when the absorptivity is one. A perfect absorber — and thus a maximal emitter — is called a *blackbody*.

The correct theoretical description of the radiation of a blackbody by *Planck* in 1900 required the assumption of emission and absorption of radiation in discrete energy quanta $E = h\nu$. The spectral radiance of a blackbody with the

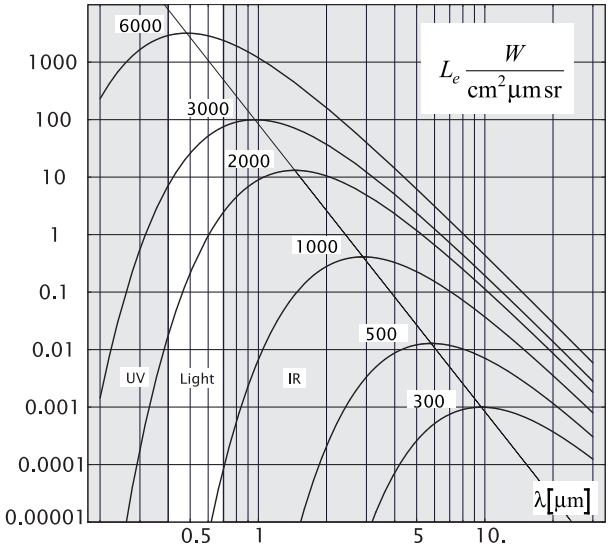

Figure 6.8: *Spectral radiance L_e of a blackbody at different absolute temperatures T in K as indicated. The thin line crosses the emission curves at the wavelength of maximum emission.*

absolute temperature T is (Fig. 6.8):

$$L_e(\nu, T) = \frac{2h\nu^3}{c^2} \frac{1}{\exp\left(\frac{h\nu}{k_B T}\right) - 1}, \quad L_e(\lambda, T) = \frac{2hc^2}{\lambda^5} \frac{1}{\exp\left(\frac{hc}{k_B T \lambda}\right) - 1}, \tag{6.24}$$

with

$$\begin{array}{ll} h = 6.6262 \times 10^{-34}\,\text{J s} & \text{Planck constant,} \\ k_B = 1.3806 \times 10^{-23}\,\text{J K}^{-1} & \text{Boltzmann constant, and} \\ c = 2.9979 \times 10^{8}\,\text{m s}^{-1} & \text{speed of light in vacuum.} \end{array} \tag{6.25}$$

Blackbody radiation has the important feature that the emitted radiation does not depend on the viewing angle. Such a radiator is called a *Lambertian radiator*. Therefore the spectral emittance (constant radiance integrated over a hemisphere) is π times higher than the radiance:

$$M_e(\lambda, T) = \frac{2\pi hc^2}{\lambda^5} \frac{1}{\exp\left(\frac{hc}{k_B T \lambda}\right) - 1}. \tag{6.26}$$

The total emittance of a blackbody integrated over all wavelengths is proportional to T^4 according to the law of Stefan and Boltzmann:

$$M_e = \int_0^\infty M_e(\lambda)\,\mathrm{d}\lambda = \frac{2}{15} \frac{k_B^4 \pi^5}{c^2 h^3} T^4 = \sigma T^4, \tag{6.27}$$

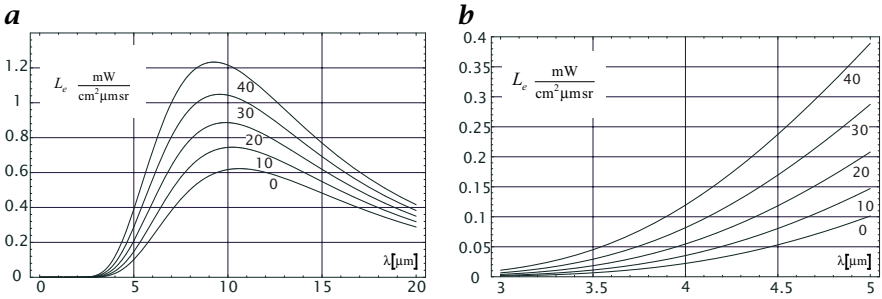

Figure 6.9: *Radiance of a blackbody at environmental temperatures as indicated in the wavelength ranges of* ***a*** *0–20 µm and* ***b*** *3–5 µm.*

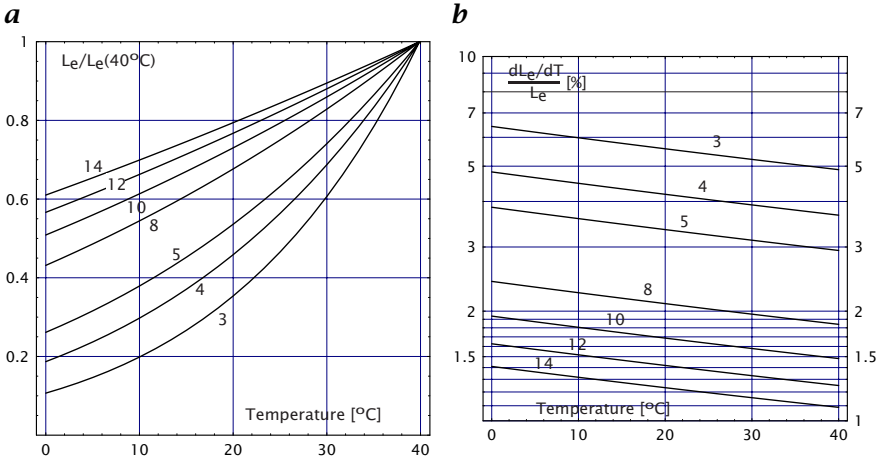

Figure 6.10: *Relative photonen-based radiance in the temperature interval 0–40°C and at wavelengths in µm as indicated:* ***a*** *related to the radiance at 40°C;* ***b*** *relative change in percent per degree.*

where $\sigma \approx 5.67 \cdot 10^{-8}\mathrm{W\,m^{-2}K^{-4}}$ is the *Stefan–Boltzmann constant*. The wavelength of maximum emittance of a blackbody is given by *Wien's law*:

$$\lambda_m \approx \frac{2.898 \cdot 10^{-3}\mathrm{K\,m}}{T}. \tag{6.28}$$

The maximum excitance at room temperature (300 K) is in the infrared at about 10 µm and at 3000 K (incandescent lamp) in the near infrared at 1 µm.

Real objects emit less radiation than a blackbody. The ratio of the emission of a real body to the emission of the blackbody is called (specific) *emissivity* ϵ and depends on the wavelength.

Radiation in the *infrared* and *microwave* range can be used to image the *temperature distribution* of objects. This application of imaging is known as *thermography*. Thermal imaging is complicated by the fact that real objects are not perfect black bodies. Thus they partly reflect radiation from the surrounding.

If an object has *emissivity* ϵ, a fraction $1-\epsilon$ of the received radiation originates from the environment, biasing the temperature measurement. Under the simplifying assumption that the environment has a constant temperature T_e, we can estimate the influence of the reflected radiation on the temperature measurement. The total radiance emitted by the object, E, is

$$E = \epsilon\sigma T^4 + (1-\epsilon)\sigma T_e^4. \tag{6.29}$$

This radiance is interpreted to originate from a blackbody with the apparent temperature T':

$$\sigma T'^4 = \epsilon\sigma T^4 + (1-\epsilon)\sigma T_e^4. \tag{6.30}$$

Rearranging for T' yields

$$T' = T\left(1 + (1-\epsilon)\frac{T_e^4 - T^4}{T^4}\right)^{1/4}. \tag{6.31}$$

In the limit of small temperature differences ($T_e - T \ll T$) Eq. (6.31) reduces to

$$T' \approx \epsilon T + (1-\epsilon)T_e \quad \text{or} \quad T' - T = (1-\epsilon)(T_e - T). \tag{6.32}$$

From this simplified equation, we infer that a 1% deviation of ϵ from unity results in 0.01 K temperature error per 1 K difference between the object temperature and the environmental temperature. Even for an almost perfect *blackbody* such as a water surface with a mean emissivity of about 0.97, this leads to considerable errors in the absolute temperature measurements. The apparent temperature of a bright sky can easily be 80 K colder than the temperature of a water surface at 300 K, leading to a $-0.03 \cdot 80\,\text{K} = -2.4\,\text{K}$ bias in the measurement of the absolute temperature of the water surface.

This bias can, according to Eqs. (6.31) and (6.32), be corrected if the mean temperature of the environment is known. Also relative temperature measurements are biased, although to a less significant degree. Assuming a constant environmental temperature in the limit $(T_e - T) \ll T$, we can infer from Eq. (6.32) that

$$\partial T' \approx \epsilon \partial T \quad \text{for} \quad (T_e - T) \ll T, \tag{6.33}$$

which means that the measured temperature differences are smaller by the factor ϵ than in reality.

Other corrections must be applied if radiation is significantly absorbed on the way from the object to the receiver. If the distance between the object and the camera is large, as for space-based or aerial infrared imaging of the Earth's surface, it is important to select a wavelength range with a minimum absorption. The two most important atmospheric windows are at 3–5 μm (with a sharp absorption peak around 4.15 μm due to CO_2) and at 8–12 μm.

Figure 6.9 shows the radiance of a blackbody at environmental temperatures between 0 and 40 °C in the 0–20 μm and 3–5 μm wavelength ranges. Although the radiance has its maximum around 10 μm and is about 20 times higher than at 4 μm, the relative change of the radiance with temperature is much larger at 4 μm than at 10 μm.

This effect can be seen in more detail by examining radiance relative to the radiance at at fixed temperature (Fig. 6.10a) and the relative radiance change in $(\partial L/\partial T)/L$ in percent (Fig. 6.10b). While the radiance at 20°C changes only about

Figure 6.11: *Some examples of thermography:* ***a*** *Heidelberg University building taken on a cold winter day,* ***b*** *street scene,* ***c*** *look inside a PC, and* ***d*** *person with lighter.*

1.7 %/K at 10 μm wavelength, it changes about 4 %/K at 4 μm wavelength. This higher relative sensitivity makes it advantageous to use the 3–5 μm wavelength range for measurements of small temperature differences although the absolute radiance is much lower.

Some images illustrating the application of *thermography* are shown in Fig. 6.11.

6.4.2 Refraction, Reflection, and Transmission‡

At the interface between two optical media, according to *Snell's law* the transmitted ray is refracted, i. e., changes direction (Fig. 6.12):

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{n_2}{n_1}, \tag{6.34}$$

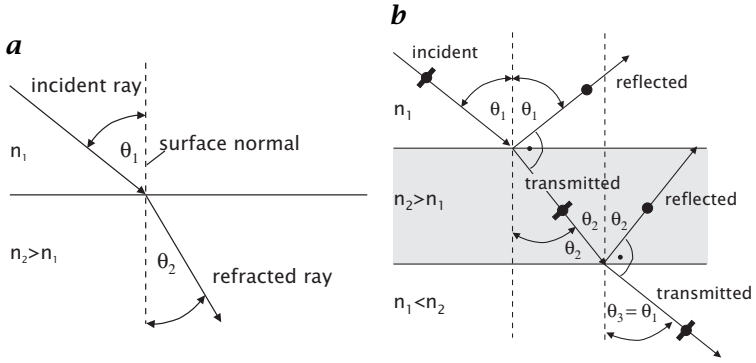

Figure 6.12: ***a*** *A ray changes direction at the interface between two optical media with a different index of refraction.* ***b*** *Parallel polarized light is entirely transmitted and not reflected when the angle between the reflected and transmitted beam would be 90°. This condition occurs at the transitions from both the optically thinner medium and the thicker one.*

where θ_1 and θ_2 are the angles of incidence and refraction, respectively. Refraction is the basis for transparent optical elements (lenses) that can form an image of an object. This means that all rays emitted from a point of the object and passing through the optical element converge at another point at the image plane.

A *specular surface* behaves like a mirror. Light irradiated in the direction (θ_i, ϕ_i) is reflected back in the direction $(\theta_i, \varphi_i + \pi)$. This means that the angle of reflectance is equal to the angle of incidence and that the incident and reflected ray and the normal of the surface lie in one plane. The ratio of the reflected radiant flux to the incident flux at the surface is called the *reflectivity* ρ.

Specular reflection only occurs when all parallel incident rays are reflected as parallel rays. A surface need not be perfectly smooth for specular reflectance because of the wave-like nature of electromagnetic radiation. It is sufficient that the residual surface irregularities are significantly smaller than the wavelength.

The reflectivity ρ depends on the angle of incidence, the refractive indices, n_1 and n_2, of the two media meeting at the interface, and the polarization state of the radiation. Light is called parallel or perpendicular polarized if the electric field vector is parallel or perpendicular to the plane of incidence, i. e., the plane containing the directions of incidence, reflection, and the surface normal.

Fresnel's equations give the reflectivity for parallel polarized light:

$$\rho_{\parallel} = \frac{\tan^2(\theta_1 - \theta_2)}{\tan^2(\theta_1 + \theta_2)}, \tag{6.35}$$

for perpendicular polarized light

$$\rho_{\perp} = \frac{\sin^2(\theta_1 - \theta_2)}{\sin^2(\theta_1 + \theta_2)}, \tag{6.36}$$

and for unpolarized light (see Fig. 6.13)

$$\rho = \frac{\rho_{\parallel} + \rho_{\perp}}{2}, \tag{6.37}$$

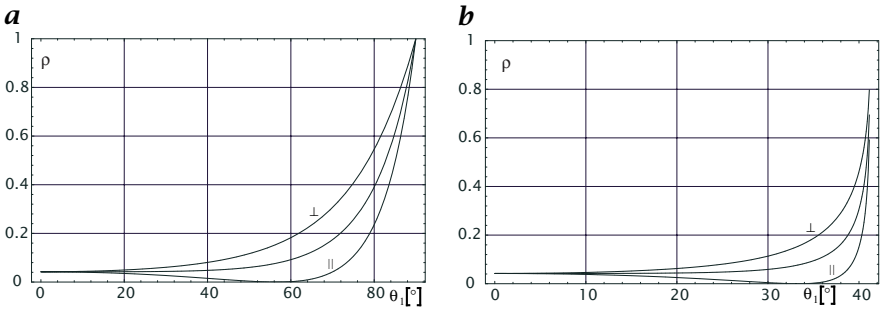

Figure 6.13: *Interface reflectivities for parallel (∥) and perpendicular (⊥) polarized light and unpolarized light incident from* ***a*** *air ($n_1 = 1.00$) to BK7 glass ($n_2 = 1.517$),* ***b*** *BK7 glass to air.*

respectively, where θ_1 and θ_2 are the angles of the incident and refracted rays related by Snell's law.

At normal incidence ($\theta_1 = 0$), the reflectivity does not depend on the polarization state:

$$\rho = \frac{(n_1 - n_2)^2}{(n_1 + n_2)^2} = \frac{(n-1)^2}{(n+1)^2} \quad \text{with} \quad n = n_1/n_2. \tag{6.38}$$

As illustrated in Fig. 6.13, parallel polarized light is not reflected at all at a certain angle, the polarizing or *Brewster angle* θ_b. This condition occurs when the refracted and reflected rays would be perpendicular to each other (Fig. 6.12b):

$$\theta_b = \arcsin \frac{1}{\sqrt{1 + n_1^2/n_2^2}}. \tag{6.39}$$

When a ray enters into a medium with lower refractive index, there is a critical angle, θ_c

$$\theta_c = \arcsin \frac{n_1}{n_2} \quad \text{with} \quad n_1 < n_2, \tag{6.40}$$

beyond which all light is reflected and none enters the optically thinner medium. This phenomenon is called *total reflection*.

6.4.3 Rough Surfaces‡

Most natural and also artificial objects do not reflect light directly but show a diffuse reflectance, as microscopic surface roughness causes reflection in various directions depending on the slope distribution of the reflecting facets. There is a great variety in how these rays are distributed over the emerging solid angle. Some materials produce strong forward scattering effects while others scatter almost equally in all directions. Other materials show a kind of mixed reflectivity which is partly specular due to reflection at the smooth surface and partly diffuse caused by body reflection. In this case, light penetrates partly into the object where it is scattered at optical inhomogeneities. Part of this scattered light leaves the object again, causing a diffuse reflection. To image objects that

do not emit radiation by themselves but passively reflect incident light, it is essential to know how the light is reflected.

Generally, the relation between the incident and emitted radiance can be expressed as the ratio of the radiance emitted at the polar angle θ_e and the azimuth angle ϕ_e and the irradiance received at the incidence angle θ_i. This ratio is called the *bidirectional reflectance distribution function* (*BRDF*) or reflectivity distribution, since it generally depends on the angles of both the incident and excitant radiance:

$$f(\theta_i, \phi_i, \theta_e, \phi_e) = \frac{L_e(\theta_e, \phi_e)}{E_i(\theta_i, \phi_i)}. \tag{6.41}$$

For a perfect mirror (specular reflection), f is zero everywhere, except for $\theta_i = \theta_e$ and $\phi_e = \pi + \phi_i$, hence

$$f(\theta_i, \theta_e) = \delta(\theta_i - \theta_e) \cdot \delta(\phi_e + \pi - \phi_i). \tag{6.42}$$

The other extreme is a perfect diffuser, reflecting incident radiation equally into all directions independently of the angle of incidence. Such a surface is known as *Lambertian* radiator or Lambertian reflector. The radiance of such a surface is independent of the viewing direction:

$$L_e = \frac{1}{\pi} E_i \quad \text{or} \quad f(\theta_i, \phi_i, \theta_e, \phi_e) = \frac{1}{\pi}. \tag{6.43}$$

6.4.4 Absorptance and Transmittance‡

Radiation traveling in matter is more or less absorbed and converted into different energy forms, especially heat. The absorptance is proportional to the radiant intensity in a thin layer $\mathrm{d}x$. Therefore

$$\frac{\mathrm{d}I(\lambda)}{\mathrm{d}x} = -\alpha(\lambda, x) I. \tag{6.44}$$

The *absorption coefficient* α is a property of the medium and depends on the wavelength of the radiation. It is a reciprocal length with the units m^{-1}. By integration of Eq. (6.44), we can compute the attenuation of radiation over the distance from 0 to x:

$$I(x) = I(0) \cdot \exp\left(-\int_0^x \alpha(\lambda, x')\mathrm{d}x'\right), \tag{6.45}$$

or, if the medium is homogeneous (i. e., α does not depend on the position x'),

$$I(x) = I(0) \exp(-\alpha(\lambda) x). \tag{6.46}$$

The exponential attenuation of radiation in a homogeneous medium, as expressed by Eq. (6.46), is often referred to as *Lambert–Beer's* or *Bouger's law*. After a distance of $1/\alpha$, the radiation is attenuated to $1/e$ of its initial value.

The path integral over the absorption coefficient

$$\tau(x_1, x_2) = \int_{x_1}^{x_2} \alpha(x')\mathrm{d}x' \tag{6.47}$$

results in a dimensionless quantity that is known as the *optical thickness* or *optical depth*. The optical depth is a logarithmic expression of radiation attenuation and means that along the path from the point x_1 to point x_2 the radiation has been attenuated to $e^{-\tau}$.

If radiation travels in a composite medium, often only one chemical species — at least at certain wavelengths — is responsible for the attenuation of the radiation. Therefore, it makes sense to relate the absorption coefficient to the concentration of that species:

$$\alpha = \varepsilon \cdot c, \quad [\varepsilon] = \left[\frac{\mathrm{l}}{\mathrm{mol\,m^{-1}}}\right], \tag{6.48}$$

where c is the concentration in mol/l. Then, ε is known as the *molar absorption coefficient*. The simple linear relation Eq. (6.48) holds for a very wide range of radiant intensities but breaks down at very high intensities, e. g., the absorption of highly intense laser beams. At that point, the domain of nonlinear optical phenomena is entered.

As the absorption coefficient is a distinct optical feature of chemical species, it can be used in imaging applications to identify chemical species and to measure their concentrations.

Finally, the term *transmittance* means the fraction of radiation that remains after the radiation has traveled a certain path in the medium. Often, transmittance and *transmissivity* are confused. In contrast to transmittance, the term transmissivity is related to a single surface. It means the fraction of radiation that is not reflected but enters the medium.

6.4.5 Scattering‡

The attenuation of radiation by scattering can be described with the same concepts as for loss of radiation by absorption. The scattering coefficient is defined by

$$\beta(\lambda) = -\frac{1}{I}\frac{\mathrm{d}I(\lambda)}{\mathrm{d}x}. \tag{6.49}$$

It is a reciprocal length with the units $\mathrm{m^{-1}}$. If in a medium the radiation is attenuated both by absorption and scattering, the two effects can be combined in the *extinction coefficient* $\kappa(\lambda)$:

$$\kappa(\lambda) = \alpha(\lambda) + \beta(\lambda). \tag{6.50}$$

Unfortunately, there is no unified terminology and symbolism for these various coefficients. The different communities use different symbols and slightly different definitions.

Although scattering appears to be similar to absorption, it is a much more difficult phenomenon. The above formula can only be used if the radiation from the individual scattering events adds up incoherently at some point far from the particles. The complexity of scattering is related to the fact that the scattered radiation (without additional absorption) is never lost. Scattered light can be scattered more than once. Therefore, a fraction of it can reenter the original beam. The probability that radiance will be scattered in a certain path length more than once is directly related to the total attenuation by scattering along

the path of the beam or the optical depth τ. If τ is smaller than 0.1, less than 10% of the radiance is scattered.

The total amount of scattered light and the analysis of the angular distribution is related to the optical properties of the scattering medium. Consequently, scattering is caused by the optical inhomogeneity of the medium. In the further discussion we assume that small spherical particles with radius r and index of refraction n are imbedded in a homogeneous optical medium.

Scattering by a particle is described by the *cross section*. It is defined in terms of the ratio of the flux removed by the particle to the flux incident on the particle:

$$\sigma_s = \phi_s / \phi \pi r^2. \tag{6.51}$$

The cross section has the units of area. It can be regarded as the effective area of the particle for scattering that completely scatters the incident radiative flux. Therefore, the *efficiency factor* for scattering Q_s is defined as the cross section related to the geometric cross section of the scattering particle:

$$Q_s = \sigma_s / (\pi r^2). \tag{6.52}$$

The angular distribution of the scattered radiation is given by the *differential cross section* $\mathrm{d}\sigma_s/\mathrm{d}\Omega$, i. e., the flux density scattered per unit solid angle. The total cross-section is given as the integral over the sphere of the differential cross-section:

$$\sigma_s = \int \frac{\mathrm{d}\sigma_s}{\mathrm{d}\Omega} \mathrm{d}\Omega. \tag{6.53}$$

The relation between the scattering coefficient β Eq. (6.49) and the scattering cross-section can be derived as follows. N is the number of particles per unit volume. Thus, the total effective scattering cross-section covers the area $N \cdot \sigma$. This area compared to the unit area gives the fraction of area that removes the incident flux and is thus equal to the scattering coefficient β:

$$\beta = N\sigma. \tag{6.54}$$

The scattering by small particles is most significantly influenced by the ratio of the particle size to the wavelength of the radiation expressed in the dimensionless particle size $q = 2\pi r/\lambda = rk$. If $q \ll 1$ (Rayleigh scattering), the scattering is very weak and proportional to λ^{-4}:

$$\sigma_s / \pi r^2 = \frac{8}{3} q^4 \left| \frac{n^2 - 1}{n^2 + 2} \right|. \tag{6.55}$$

For $q \gg 1$, the scattering can be described by geometrical optics. If the particle completely reflects the incident radiation, the scattering cross-section is equal to the geometric cross-section ($\sigma_s/\pi r^2 = 1$) and the differential cross-section is constant (isotropic scattering, $\mathrm{d}\sigma/\mathrm{d}\Omega = r^2/2$).

Scattering for particles with sizes of about the wavelength of the radiation (*Mie scattering*) is very complex due to diffraction and interference effects of the light scattered from different portions of the surface of the particle. The differential cross-section shows strong variations with the scattering angle and is directed mostly in the forward direction, while Rayleigh scattering is fairly isotropic.

6.4.6 Optical Activity‡

An optical material rotates the plane of polarization of electromagnetic radiation. The rotation is proportional to the concentration of the *optically active* material, c, and the path length d:

$$\varphi = \gamma(\lambda) c d. \tag{6.56}$$

The constant γ is known as the *specific rotation* and has the units [m^2 mol] or [cm^2 g^{-1}]; it depends strongly on the wavelength of the radiation. Generally, the specific rotation is significantly larger at shorter wavelengths.

Two well-known optically active materials are quartz crystals and sugar solution. Optical activity — including the measurement of the wavelength dependency — can be used to identify chemical species and to measure their concentration. With respect to visualization, optical activity is significant since it can be induced by various external forces, among others electrical fields (*Kerr effect*) and magnetic fields (*Faraday effect*).

6.4.7 Luminescence‡

Luminescence is the emission of radiation from materials that arises from a radiative transition from an excited state to a lower state. *Fluorescence* is luminescence characterized by short lifetimes of the excited state (on the order of nanoseconds), while the term *phosphorescence* is used for longer lifetimes (milliseconds to minutes).

Luminescence is an enormously versatile process since it can be triggered by various processes. In *chemiluminescence*, the energy required to generate the excited state is derived from the energy released by a chemical reaction. Chemiluminescence normally has only low efficiencies (i. e., number of photons emitted per reacting molecule) on the order of 1 % or less. Flames are the classic example of a low-efficiency chemiluminescent process. *Bioluminescence* is a chemiluminescence in living organisms. Fireflies and the glow of marine microorganisms are well-known examples. The firefly reaction involves the enzymatic oxidation of luciferin. In contrast to most chemiluminescent processes, this reaction converts almost 100 % of the chemical energy into radiant energy. Low-level bioluminescent processes are common to many essential biological processes. Imaging of these processes is becoming an increasingly important tool to study biochemical processes.

Marking biomolecules with fluorescent dyes is becoming another increasingly sophisticated tool in biochemistry. It has even become possible to mark individual chromosomes or gene sequences in chromosomes with fluorescent dyes.

Luminescence always has to compete with other processes that deactivate the excited state without radiation emission. A prominent radiationless deactivation process is the energy transfer during the collision of molecules.

Some types of molecules, especially electronegative molecules such as *oxygen*, are very efficient in deactivating excited states during collisions. This process is referred to by the term *quenching*. The presence of a quenching molecule causes the fluorescence to decrease. Therefore, the measurement of the fluorescent irradiance can be used to measure the concentration of the quenching

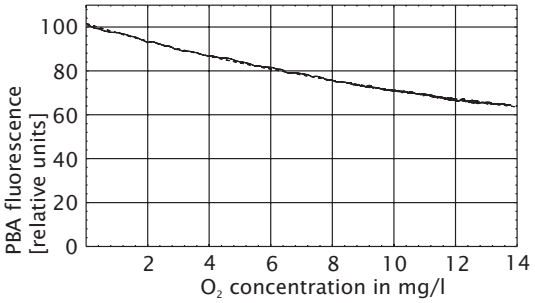

Figure 6.14: *Quenching of the fluorescence of pyrene butyric acid by dissolved oxygen: measurements and fit with the Stern–Vollmer equation (dashed line).*

molecule. The dependence of the fluorescent intensity on the concentration of the quencher is given by the *Stern–Vollmer equation*:

$$\frac{L}{L_0} = \frac{1}{1 + kc_q}, \tag{6.57}$$

where L is the fluorescent radiance, L_0 the fluorescent radiance when no quencher is present, C_q the quencher concentration, and k the quenching constant depending suitably on the lifetime of the fluorescent state. Efficient quenching requires that the excited state have a sufficiently long lifetime.

A fluorescent dye suited for quenching by dissolved oxygen is *pyrene butyric acid* (*PBA*) [189]. The relative fluorescent radiance of PBA as a function of dissolved oxygen is shown in Fig. 6.14 [127]. Fluorescence is stimulated by a pulsed nitrogen laser at 337 nm. The change in fluorescence is rather weak but sufficiently large to enable reliable measurements of the concentration of dissolved oxygen.

6.4.8 Doppler Effect‡

A velocity difference between a radiating source and a receiver causes the receiver to measure a frequency different from that emitted by the source. This phenomenon is known as the *Doppler effect*. The frequency shift is directly proportional to the velocity difference according to

$$\nu_r = \frac{c - \boldsymbol{u}_r^T \bar{\boldsymbol{k}}}{c - \boldsymbol{u}_s^T \bar{\boldsymbol{k}}} \nu_s \quad \text{or} \quad \Delta\nu = \nu_r - \nu_s = \frac{(\boldsymbol{u}_s - \boldsymbol{u}_r)^T \boldsymbol{k}}{1 - \boldsymbol{u}_s^T \bar{\boldsymbol{k}}/c}, \tag{6.58}$$

where $\bar{\boldsymbol{k}} = \boldsymbol{k}/|\boldsymbol{k}|$, ν_s is the frequency of the source, ν_r the frequency measured at the receiver, $\boldsymbol{k}$ the wave number of the radiation, c the propagation speed of the radiation, and $\boldsymbol{u}_s$ and $\boldsymbol{u}_r$ the velocities of the source and receiver relative to the medium in which the wave is propagating. Only the velocity component in the direction to the receiver causes a frequency shift.

If the source is moving towards the receiver ($\boldsymbol{u}_s \boldsymbol{k} > 0$), the frequency increases as the wave fronts follow each other faster. A critical limit is reached when the

source moves with the propagation speed of the radiation. Then, the radiation is left behind the source. For small velocities relative to the wave propagation speed, the frequency shift is directly proportional to the relative velocity between source and receiver.

$$\Delta\nu = (\boldsymbol{u}_s - \boldsymbol{u}_r)\boldsymbol{k}. \tag{6.59}$$

The relative frequency shift $\Delta\omega/\omega$ is given directly by the ratio of the velocity difference in the direction of the receiver and the wave propagation speed:

$$\frac{\Delta\nu}{\nu} = \frac{(\boldsymbol{u}_s - \boldsymbol{u}_r)^T}{c}\bar{\boldsymbol{k}}. \tag{6.60}$$

For electromagnetic waves, the velocity relative to a "medium" is not relevant. The theory of relativity gives the frequency

$$\nu_r = \frac{\nu_s}{\gamma(1 - \boldsymbol{u}^T\bar{\boldsymbol{k}}/c)} \quad \text{with} \quad \gamma = \frac{1}{\sqrt{1 - (|\boldsymbol{u}|/c)^2}}. \tag{6.61}$$

For small velocities ($|\boldsymbol{u}| << c$), this equation also reduces to Eq. (6.59) with $\boldsymbol{u} = \boldsymbol{u}_s - \boldsymbol{u}_r$. In this case, acoustic and electromagnetic waves can be treated equally with respect to the frequency shift due to a relative velocity between the source and receiver.

6.5 Further Readings‡

This chapter covered a variety of topics that are not central to image processing yet are important to know for a correct image acquisition. You can refresh or extend your knowledge about electromagnetic waves by one of the classical textbooks on the subject, e. g., F. S. Crawford [36], Hecht [66], or Towne [184]. Stewart [178] and Drury [33] address the interaction of radiation with matter in the field of remote sensing. Richards [150] gives a survey of imaging techniques across the electromagnetic spectrum. The topic of infrared imaging has become an subject of its own and is treated in detail by Gaussorgues [49] and Holst [71]. Pratt [142] give a good description of color vision with respect to image processing. The practical aspects of photometry and radiometry are covered by the "Handbook of Applied Photometry" from DeCusaris [26]. The oldest application area of quantitative visualization is hydrodynamics. A fascinating insight into flow visualization with many images is given by the "Atlas of Visualization" edited by Nakayama and Tanida [129].

7 Image Formation

7.1 Introduction

Image formation includes three major aspects. One is *geometric* in nature. The question is *where* we find an object in the image. Essentially, all imaging techniques project a three-dimensional space in one way or the other onto a two-dimensional image plane. Thus, basically, imaging can be regarded as a projection from 3-D into 2-D space. The loss of one coordinate constitutes a severe loss of information about the geometry of the observed scene. However, we unconsciously and constantly experience that our visual system perceives a three-dimensional impression sufficiently well that we can grasp the three-dimensional world around us and interact with it. The ease with which this reconstruction task is performed by biological visual systems might tempt us to think that this is a simple task. But — as we will see in Chapters 8 and 17 — it is not that simple.

The second aspect is *radiometric* in nature. How "bright" is an imaged object, and how does the brightness in the image depend on the optical properties of the object and the image formation system? The radiometry of an imaging system is discussed in Section 7.5. For the basics of radiometry see Section 6.3.

The third question is, finally, what happens to an image when we represent it with an array of digital numbers to process it with a digital computer? How do the processes that transform a continuous image into such an array — known as *digitization* and *quantization* — limit the resolution in the image or introduce artifacts? These questions are addressed in Chapter 9.

7.2 World and Camera Coordinates

7.2.1 Definition

Basically, the position of objects in 3-D space can be described in two different ways (Fig. 7.1). First, we can use a coordinate system which is related to the scene observed. These coordinates are called *world coordinates* and denoted as $\boldsymbol{X}' = [X'_1, X'_2, X'_3]^T$. The X'_1 and X'_2 coordinates describe the horizontal and X'_3 the vertical positions, respec-

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2

Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

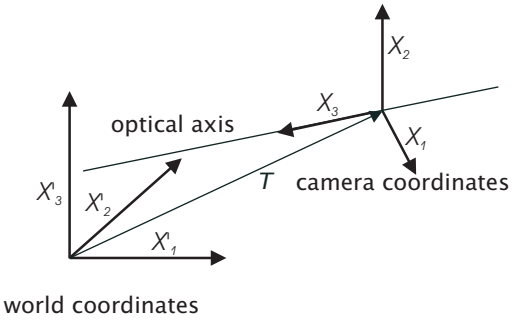

Figure 7.1: *Illustration of world and camera coordinates.*

tively. Sometimes, an alternative convention with non-indexed coordinates $\boldsymbol{X}' = [X', Y', Z']^T$ is more convenient. Both notations are used in this book.

A second system, the *camera coordinates* $\boldsymbol{X} = [X_1, X_2, X_3]^T$, can be fixed to the camera observing the scene. The X_3 axis is aligned with the *optical axis* of the camera system (Fig. 7.1). Physicists are familiar with such considerations. It is common to discuss physical phenomena in different coordinate systems. In elementary mechanics, for example, motion is studied with respect to two observers, one at rest, the other moving with the object.

Transition from world to camera coordinates generally requires a *translation* and a *rotation*. First, we shift the origin of the world coordinate system to the origin of the camera coordinate system by the translation vector $\boldsymbol{T}$ (Fig. 7.1). Then we change the orientation of the shifted system by rotations about suitable axes so that it coincides with the camera coordinate system. Mathematically, translation can be described by vector subtraction and rotation by multiplication of the coordinate vector with a matrix:

$$\boldsymbol{X} = \boldsymbol{R}(\boldsymbol{X}' - \boldsymbol{T}). \tag{7.1}$$

7.2.2 Rotation

Rotation of a coordinate system has two important features. It does not change the length or *norm* of a vector and it keeps the coordinate system orthogonal. A transformation with these features is known in linear algebra as an *orthonormal* transform.

The coefficients in a transformation matrix have an intuitive meaning. This can be seen when we apply the transformation to unit vectors $\bar{\boldsymbol{E}}_p$

in the direction of the coordinate axes. With $\bar{\boldsymbol{E}}_1$, for instance, we obtain

$$\bar{\boldsymbol{E}}_1' = \boldsymbol{A}\bar{\boldsymbol{E}}_1 = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \end{bmatrix}. \tag{7.2}$$

Thus, the columns of the transformation matrix give the coordinates of the base vectors in the new coordinate system. Knowing this property, it is easy to formulate the orthonormality condition that has to be met by the rotation matrix R:

$$\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I} \quad \text{or} \quad \sum_{m=1}^{3} r_{km}r_{lm} = \delta_{k-l}, \tag{7.3}$$

where $\boldsymbol{I}$ denotes the identity matrix, whose elements are one and zero on diagonal and non-diagonal positions, respectively. Using Eq. (7.2), this equation simply states that the transformed base vectors remain orthogonal:

$$\bar{\boldsymbol{E}}_k'^T\bar{\boldsymbol{E}}_l' = \delta_{k-l}. \tag{7.4}$$

Equation Eq. (7.3) leaves three matrix elements independent out of nine. Unfortunately, the relationship between the matrix elements and three parameters to describe rotation turns out to be quite complex and nonlinear. A common procedure involves the three Eulerian rotation angles (ϕ, θ, ψ). A lot of confusion exists in the literature about the definition of the Eulerian angle. We follow the standard mathematical approach. We use right-hand coordinate systems and count rotation angles positive in the counterclockwise direction. The rotation from the shifted world coordinate system to the camera coordinate system is decomposed into three steps (see Fig. 7.2, [53]).

1. Rotation about X_3' axis by the angle ϕ, $\boldsymbol{X}'' = \boldsymbol{R}_\phi\boldsymbol{X}'$:

$$\boldsymbol{R}_\phi = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.5}$$

2. Rotation about X_1'' axis by θ, $\boldsymbol{X}''' = \boldsymbol{R}_\theta\boldsymbol{X}''$:

$$\boldsymbol{R}_\theta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \tag{7.6}$$

3. Rotation about X_3''' axis by ψ, $\boldsymbol{X} = \boldsymbol{R}_\psi\boldsymbol{X}'''$:

$$\boldsymbol{R}_\psi = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.7}$$

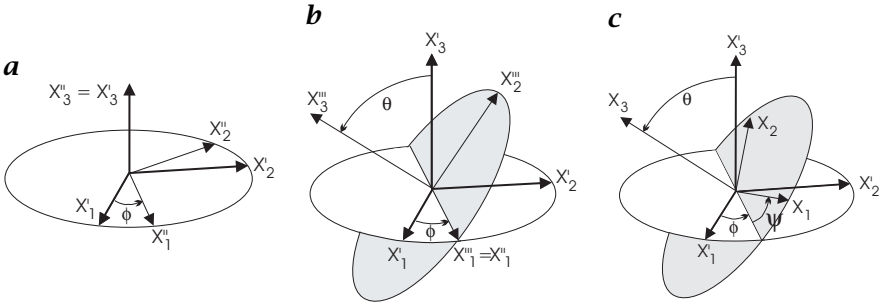

Figure 7.2: *Rotation of world coordinates $\boldsymbol{X}'$ to camera coordinates $\boldsymbol{X}$ using the three Eulerian angles (ϕ, θ, ψ) with successive rotations about the* ***a*** X_3', ***b*** X_1'', *and* ***c*** X_3''' *axes.*

Cascading the three rotations, $\boldsymbol{R}_\psi \boldsymbol{R}_\theta \boldsymbol{R}_\phi$, yields the matrix

$$\begin{bmatrix} \cos\psi\cos\phi - \cos\theta\sin\phi\sin\psi & \cos\psi\sin\phi + \cos\theta\cos\phi\sin\psi & \sin\theta\sin\psi \\ -\sin\psi\cos\phi - \cos\theta\sin\phi\cos\psi & -\sin\psi\sin\phi + \cos\theta\cos\phi\cos\psi & \sin\theta\cos\psi \\ \sin\theta\sin\phi & -\sin\theta\cos\phi & \cos\theta \end{bmatrix}.$$

The inverse transformation from camera coordinates to world coordinates is given by the transpose of the above matrix. Since matrix multiplication is not commutative, rotation is also not commutative. Therefore, it is important not to interchange the order in which rotations are performed.

Rotation is only commutative in the limit of an infinitesimal rotation. Then, the cosine and sine terms reduce to 1 and ε, respectively. This limit has some practical applications since minor rotational misalignments are common.

Rotation about the X_3 axis, for instance, can be

$$\boldsymbol{X} = \boldsymbol{R}_\varepsilon \boldsymbol{X}' = \begin{bmatrix} 1 & \varepsilon & 0 \\ -\varepsilon & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{X}' \quad \text{or} \quad \begin{aligned} X_1 &= X_1' + \varepsilon X_2' \\ X_2 &= X_2' - \varepsilon X_1' \\ X_3 &= X_3' \end{aligned}.$$

As an example we discuss the rotation of the point $[X_1', 0, 0]^T$. It is rotated into the point $[X_1', \varepsilon X_1', 0]^T$ while the correct position would be $[X_1' \cos\varepsilon, X_1' \sin\varepsilon, 0]^T$. Expanding the trigonometric function in a Taylor series to third order yields a position error of $[1/2\varepsilon^2 X_1', 1/6\varepsilon^3 X_1', 0]^T$. For a 512×512 image ($X_1' < 256$ for centered rotation) and an error limit of less than 1/20 pixel, ε must be smaller than 0.02 or 1.15°. This is still a significant rotation vertically displacing rows by up to $\pm\varepsilon X' = \pm 5$ pixels.

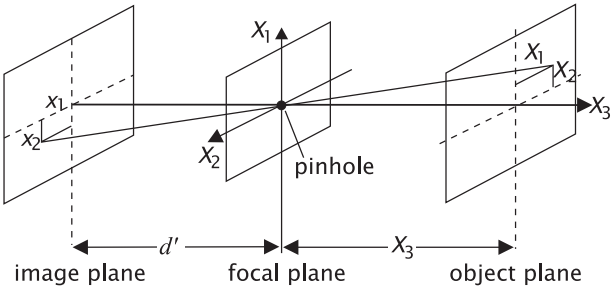

Figure 7.3: *Image formation with a pinhole camera.*

7.3 Ideal Imaging: Perspective Projection†

7.3.1 The Pinhole Camera

The basic geometric aspects of image formation by an optical system are well modeled by a *pinhole camera*. The imaging element of this camera is an infinitesimally small hole (Fig. 7.3). The single light ray coming from a point of the object at $[X_1, X_2, X_3]^T$ which passes through this hole meets the image plane at $[x_1, x_2, -d_i]^T$. Through this condition an image of the object is formed on the image plane. The relationship between the 3-D world and the 2-D *image coordinates* $[x_1, x_2]^T$ is given by

$$x_1 = -\frac{d' X_1}{X_3}, \quad x_2 = -\frac{d' X_2}{X_3}. \tag{7.8}$$

The two world coordinates parallel to the image plane are scaled by the factor d'/X_3. Therefore, the image coordinates $[x_1, x_2]^T$ contain only ratios of world coordinates, from which neither the distance nor the true size of an object can be inferred.

A straight line in the world space is projected onto a straight line at the image plane. This important feature can be proved by a simple geometric consideration. All light rays emitted from a straight line pass through the pinhole. Consequently they all lie on a plane which is spanned by the straight line and the pinhole. This plane intersects with the image plane in a straight line.

All object points on a ray through the pinhole are projected onto a single point in the image plane. In a scene with several transparent objects, the objects are projected onto each other. Then we cannot infer the three-dimensional structure of the scene at all. We may not even be able to recognize the shape of individual objects. This example demonstrates how much information is lost by projection of a 3-D scene onto a 2-D image plane.

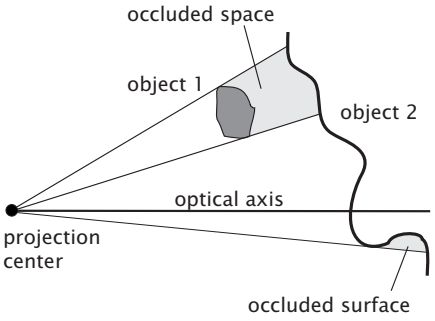

Figure 7.4: *Occlusion of more distant objects and surfaces by perspective projection.*

Most natural scenes, however, contain opaque objects. Here the observed 3-D space is essentially reduced to 2-D surfaces. These surfaces can be described by two two-dimensional functions $g(x_1, x_2)$ and $X_3(x_1, x_2)$ instead of the general description of a 3-D scalar gray value image $g(X_1, X_2, X_3)$. A surface in space is completely projected onto the image plane provided that not more than one point of the surface lies on the same ray through the pinhole. If this condition is not met, parts of the surface remain invisible. This effect is called *occlusion*. The occluded 3-D space can be made visible if we put a point light source at the position of the pinhole (Fig. 7.4). Then the invisible parts of the scene lie in the shadow of those objects which are closer to the camera.

As long as we can exclude occlusion, we only need the depth map $X_3(x_1, x_2)$ to reconstruct the 3-D shape of a scene completely. One way to produce it — which is also used by our visual system — is by stereo imaging, i. e., observation of the scene with two sensors from different points of view (Section 8.2.1).

7.3.2 Projective Imaging

Imaging with a pinhole camera is essentially a *perspective projection*, because all rays must pass through one central point, the pinhole. Thus the pinhole camera model is very similar to imaging with penetrating rays, such as x-rays, emitted from a point source (Fig. 7.5). In this case, the object lies between the central point and the image plane.

The projection equation corresponds to Eq. (7.8) except for the sign:

$$\begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \longmapsto \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{d' X_1}{X_3} \\ \frac{d' X_2}{X_3} \end{bmatrix}. \tag{7.9}$$

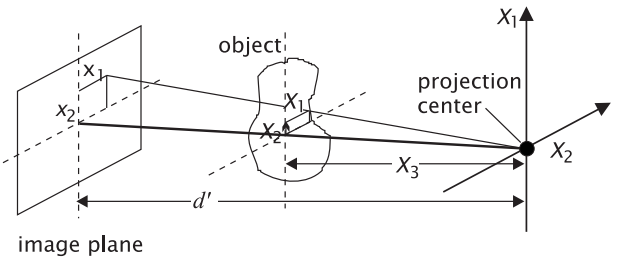

Figure 7.5: *Perspective projection with x-rays.*

The image coordinates divided by the image distance d_i are called *generalized image coordinates*:

$$\tilde{x}_1 = \frac{x_1}{d'}, \quad \tilde{x}_2 = \frac{x_2}{d'}. \tag{7.10}$$

Generalized image coordinates are dimensionless and denoted by a tilde. They are equal to the tangent of the angle with respect to the optical axis of the system with which the object is observed. These coordinates explicitly take the limitations of the projection onto the image plane into account. From these coordinates, we cannot infer absolute positions but know only the angle at which the object is projected onto the image plane. The same coordinates are used in astronomy. The general projection equation of perspective projection Eq. (7.9) then reduces to

$$\boldsymbol{X} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \longmapsto \tilde{\boldsymbol{x}} = \begin{bmatrix} \frac{X_1}{X_3} \\ \frac{X_2}{X_3} \end{bmatrix}. \tag{7.11}$$

We will use this simplified projection equation in all further considerations. For optical imaging, we just have to include a minus sign or, if speaking geometrically, reflect the image at the origin of the coordinate system.

7.3.3 Homogeneous Coordinates‡

In computer graphics, the elegant formalism of *homogeneous coordinates* [37, 46, 122] is used to describe all the transformations we have discussed so far, i. e., translation, rotation, and perspective projection, in a unified framework. This formalism is significant, because the whole image formation process can be expressed by a single 4×4 matrix.

Homogeneous coordinates are represented by a four-component column vector

$$\boldsymbol{X}' = [tX_1', tX_2', tX_3', t]^T, \tag{7.12}$$

from which ordinary three-dimensional coordinates are obtained by dividing the first three components of the homogeneous coordinates by the fourth. Any arbitrary transformation can be obtained by premultiplying the homogeneous coordinates with a 4×4 matrix $\boldsymbol{M}$. In particular, we can obtain the image coordinates

$$\boldsymbol{x} = [sx_1, sx_2, sx_3, s]^T \tag{7.13}$$

by

$$\boldsymbol{x} = \boldsymbol{M}\boldsymbol{X}. \tag{7.14}$$

Since matrix multiplication is associative, we can view the matrix $\boldsymbol{M}$ as composed of many transformation matrices, performing such elementary transformations as *translation*, *rotation* around a coordinate axis, *perspective projection*, and *scaling*. The transformation matrices for the elementary transforms are readily derived:

$$\begin{aligned}
\boldsymbol{T} &= \begin{bmatrix} 1 & 0 & 0 & T_1 \\ 0 & 1 & 0 & T_2 \\ 0 & 0 & 1 & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} && \text{Translation by } [T_1, T_2, T_3]^T \\
\boldsymbol{R}_{x_1} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} && \text{Rotation about } X_1 \text{ axis by } \theta \\
\boldsymbol{R}_{x_2} &= \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} && \text{Rotation about } X_2 \text{ axis by } \phi \\
\boldsymbol{R}_{x_3} &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 & 0 \\ \sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} && \text{Rotation about } X_3 \text{ axis by } \psi \\
\boldsymbol{S} &= \begin{bmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} && \text{Scaling} \\
\boldsymbol{P} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d' & 1 \end{bmatrix} && \text{Perspective projection.}
\end{aligned} \tag{7.15}$$

Perspective projection is formulated slightly differently from the definition in Eq. (7.11). Premultiplication of the homogeneous vector

$$\boldsymbol{X} = [tX_1, tX_2, tX_3, t]^T$$

with $\boldsymbol{P}$ yields

$$\left[tX_1, tX_2, tX_3, t\frac{d' - X_3}{d'}\right]^T, \tag{7.16}$$

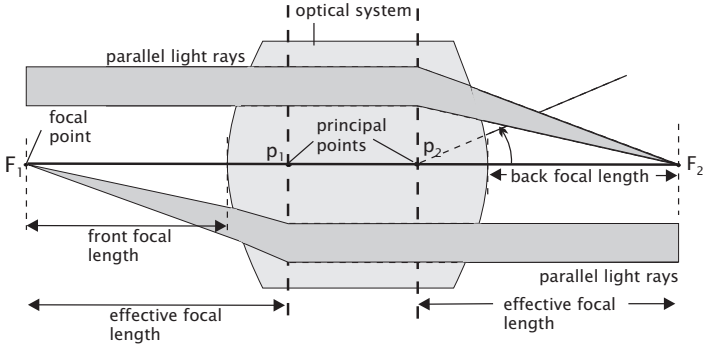

Figure 7.6: *Black box model of an optical system.*

from which we obtain the image coordinates by division through the fourth coordinate

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} X_1 \dfrac{d'}{d' - X_3} \\ X_2 \dfrac{d'}{d' - X_3} \end{bmatrix}. \tag{7.17}$$

From this equation we can see that the image plane is positioned at the origin, since if $X_3 = 0$, both image and world coordinates are identical. The center of projection has been shifted to $[0, 0, -d']^T$.

Complete transformations from world coordinates to image coordinates can be composed of these elementary matrices. Strat [179], for example, proposed the following decomposition:

$$\boldsymbol{M} = \boldsymbol{CSPR}_z\boldsymbol{R}_y\boldsymbol{R}_x\boldsymbol{T}. \tag{7.18}$$

The scaling $\boldsymbol{S}$ and cropping (translation) $\boldsymbol{C}$ are transformations taking place in the two-dimensional image plane. Strat [179] shows how the complete transformation parameters from camera to world coordinates can be determined in a noniterative way from a set of calibration points whose positions in the space are exactly known. In this way, an absolute calibration of the outer camera parameters position and orientation and the inner parameters piercing point of the optical axis, focal length, and pixel size can be obtained.

7.4 Real Imaging

7.4.1 Basic Geometry of an Optical System

The model of a pinhole camera is an oversimplification of an imaging system. A pinhole camera forms an image of an object at *any* distance while a real optical system forms a sharp image only within a certain distance range. Fortunately, the geometry even for complex optical systems can still be modeled with a small modification of perspective projection

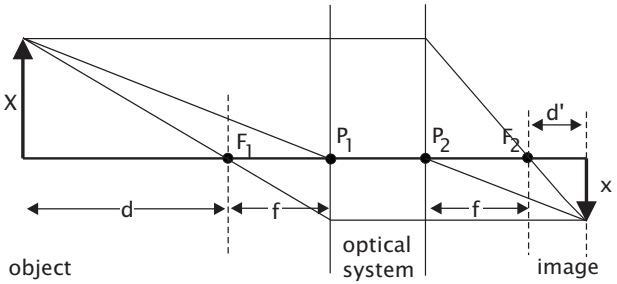

Figure 7.7: *Optical imaging with an optical system modeled by its principal points P_1 and P_2 and focal points F_1 and F_2. The system forms an image that is a distance d' behind F_2 from an object that is the distance d in front of F_1.*

as illustrated in Figs. 7.6 and 7.7. The focal plane has to be replaced by two *principal planes*. The two principal planes meet the *optical axis* at the *principal points*. A ray directed towards the first principal point appears — after passing through the system — to originate from the second principal point without angular deviation (Fig. 7.6). The distance between the two principal planes thus models the axial extension of the optical system.

As illustrated in Fig. 7.6, rays between the two principal planes are always parallel and parallel rays entering the optical system from left and right meet at the second and first focal point, respectively. For practical purposes, the following definitions also are useful: The *effective focal length* is the distance from the principal point to the corresponding focal point. The *front focal length* and *back focal length* are the distances from the first and last surface of the optical system to the first and second focal point, respectively.

The relation between the object distance and the image distance becomes very simple if they are measured from the focal points (Fig. 7.7),

$$dd' = f^2. \tag{7.19}$$

This is the Newtonian form of the *image equation*. The possibly better known Gaussian form uses the distances as to the principal points:

$$\frac{1}{d'+f} + \frac{1}{d+f} = \frac{1}{f} \tag{7.20}$$

7.4.2 Lateral and Axial Magnification

The *lateral magnification* m_l of an optical system is given by the ratio of the image size, x, to the object size, X:

$$m_l = \frac{x_1}{X_1} = \frac{f}{d} = \frac{d'}{f}. \tag{7.21}$$

A less well-known quantity is the *axial magnification* that relates the positions of the image plane and object plane to each other. Thus the axial magnification gives the magnification along the optical axis. If we shift a point in the object space along the optical axis, how large is the shift of the image plane? In contrast to the lateral magnification, the axial magnification is not constant with the position along the optical axis. Therefore the axial magnification is only defined in the limit of small changes. We use slightly modified object and image positions $d + \Delta X_3$ and $d' - \Delta x_3$ and introduce them into Eq. (7.19). Then a first-order Taylor expansion in ΔX_3 and Δx_3 (assuming that $\Delta X_3 \ll d$ and $\Delta x_3 \ll d'$) yields

$$\frac{\Delta x_3}{\Delta X_3} \approx \frac{d'}{d} \tag{7.22}$$

and the axial magnification m_a is given by

$$m_a \approx \frac{d'}{d} = \frac{f^2}{d^2} = \frac{d'^2}{f^2} = m_l^2. \tag{7.23}$$

7.4.3 Depth of Focus and Depth of Field

The image equations Eqs. (7.19) and (7.20) determine the relation between object and image distances. If the image plane is slightly shifted or the object is closer to the lens system, the image is not rendered useless. It rather gets blurred. The degree of blurring depends on the deviation from the distances given by the image equation.

The concepts of depth of focus and depth of field are based on the fact that a certain degree of blurring does not affect the image quality. For digital images it is naturally given by the size of the sensor elements. It makes no sense to resolve smaller structures. We compute the blurring in the framework of geometrical optics using the image of a point object as illustrated in Fig. 7.8a. At the image plane, the point object is imaged to a point. It smears to a disk with the radius ϵ with increasing distance from the image plane. Introducing the f-number n_f of an optical system as the ratio of the focal length and diameter of lens aperture $2r$:

$$n_f = \frac{f}{2r} \tag{7.24}$$

we can express the radius of the blur disk as:

$$\epsilon = \frac{1}{2n_f}\frac{f}{f + d'}\Delta x_3, \tag{7.25}$$

where Δx_3 is the distance from the (focused) image plane. The range of positions of the image plane, $[d' - \Delta x_3, d' + \Delta x_3]$, for which the radius of the blur disk is lower than ϵ, is known as the *depth of focus*.

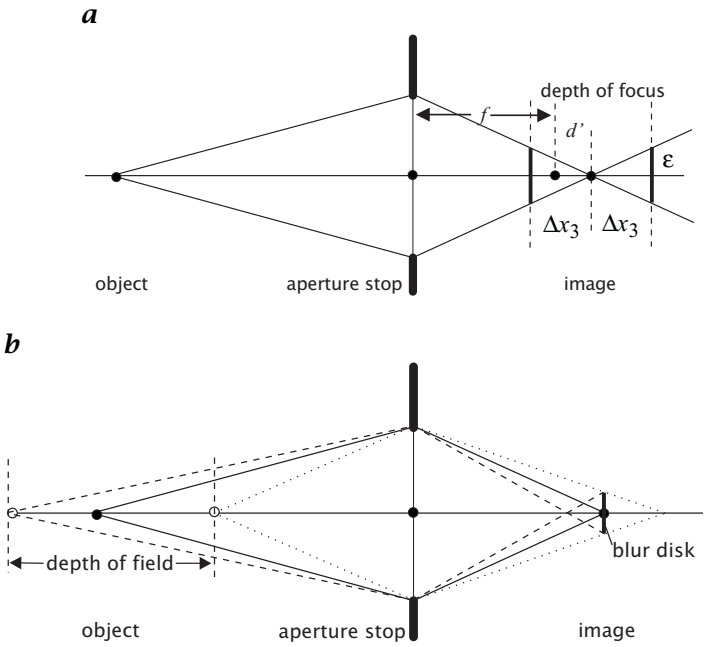

Figure 7.8: *Illustration of the* ***a*** *depth of focus and* ***b*** *depth of field with an on-axis point object.*

Equation (7.25) can be solved for Δx_3 and yields

$$\Delta x_3 = 2n_f \left(1 + \frac{d'}{f}\right) \epsilon = 2n_f(1 + m_l)\epsilon, \tag{7.26}$$

where m_l is the lateral magnification as defined by Eq. (7.21). Equation (7.26) illustrates the critical role of the n_f-number and magnification for the depth of focus. Only these two parameters determine for a given ϵ the depth of focus and depth of field.

Of even more importance for practical usage than the depth of focus is the *depth of field*. The depth of field is the range of object positions for which the radius of the blur disk remains below a threshold ϵ at a fixed image plane (Fig. 7.8b). With Eqs. (7.19) and (7.26) we obtain

$$d \pm \Delta X_3 = \frac{f^2}{d' \mp \Delta x_3} = \frac{f^2}{d' \mp 2n_f(1 + m_l)\epsilon}. \tag{7.27}$$

In the limit of $\Delta X_3 \ll d$, Eq. (7.27) reduces to

$$\Delta X_3 \approx 2n_f \cdot \frac{1 + m_l}{m_l^2}\epsilon. \tag{7.28}$$

If the depth of field includes the infinite distance, the minimum distance for a sharp image is

$$d_{\min} = \frac{f^2}{4n_f(1+m_l)\epsilon} \approx \frac{f^2}{4n_f\epsilon}. \tag{7.29}$$

A typical high resolution CCD camera has sensor elements, which are about $10 \times 10\,\mu$m in size. Thus we can allow for a radius of the unsharpness disc of $5\,\mu$m. Assuming a lens with an f-number of 2 and a focal length of 15 mm, according to Eq. (7.28) we have a depth of field of $\pm$ 0.2 m at an object distance of 1.5 m, and according to Eq. (7.29) the depth of field reaches from 5 m to infinity. This example illustrates that even with this small f-number and the relatively short distance, we may obtain a large depth of field.

For high magnifications as in *microscopy*, the depth of field is very small. With $m_l \gg 1$, Eq. (7.28) reduces to

$$\Delta X_3 \approx \frac{2n_f\epsilon}{m_l}. \tag{7.30}$$

With a 50-fold enlargement ($m_l = 50$) and $n_f = 1$, we obtain the extreme low depth of field of only $0.2\,\mu$m.

Generally, the whole concept of depth of field and depth of focus as discussed here is only valid in the limit of geometrical optics. It can only be used for blurring that is significantly larger than that caused by the aberrations or diffraction of the optical system.

7.4.4 Telecentric Imaging

In a standard optical system, a converging beam of light enters an optical system. This setup has a significant disadvantage for optical gauging (Fig. 7.9a). The object appears larger if it is closer to the lens and smaller if it is farther away from the lens. As the depth of the object cannot be inferred from its image, either the object must be at a precisely known depth or measurement errors are unavoidable.

A simple change in the position of the *aperture stop* from the principal point to the first focal point solves the problem and changes the imaging system to a telecentric lens (Fig. 7.9b). By placing the stop at this point, the *principal rays* (ray passing through the center of the aperture) are parallel to the optical axis in the object space. Therefore, slight changes in the position of the object do not change the size of the image of the object. The farther it is away from the focused position, the more it is blurred, of course. However, the center of the blur disk does not change the position.

Telecentric imaging has become an important principle in machine vision. Its disadvantage is, of course, that the diameter of a telecentric

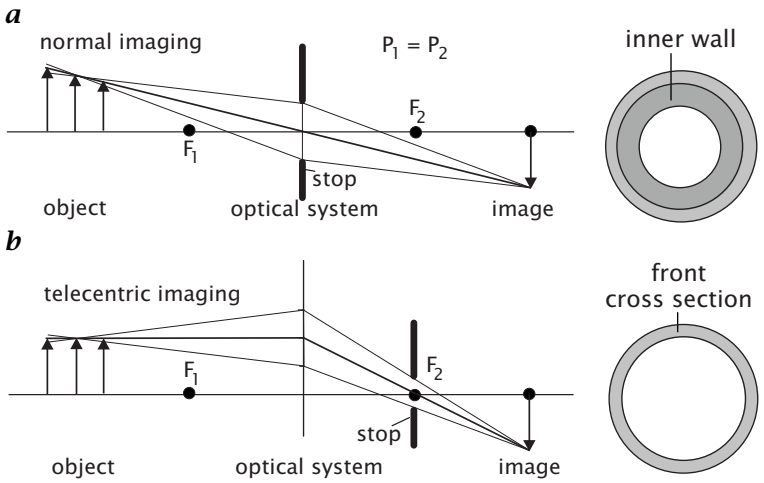

Figure 7.9: ***a*** *Standard diverging imaging with stop at the principal point;* ***b*** *telecentric imaging with stop at the second focal point. On the right side it is illustrated how a short cylindrical tube whose axis is aligned with the optical axis is imaged with the corresponding set up.*

lens must be at least of the size of the object to be imaged. This makes telecentric imaging very expensive for large objects.

Figure 7.9 illustrates how a cylinder aligned with the optical axis with a thin wall is seen with a standard lens and a telecentric lens. Standard imaging sees the cross-section and the inner wall and telecentric imaging the cross-section only.

The discussion of telecentric imaging emphasizes the importance of stops in the construction of optical systems, a fact that is often not adequately considered.

7.4.5 Geometric Distortion

A real optical system causes deviations from a perfect perspective projection. The most obvious *geometric distortions* can be observed with simple spherical lenses as barrel- or cushion-shaped images of squares. Even with a corrected lens system these effects are not completely suppressed.

This type of distortion can easily be understood by considerations of symmetry. As lens systems show cylindrical symmetry, concentric circles only suffer a distortion in the radius. This distortion can be approximated by

$$\boldsymbol{x}' = \frac{\boldsymbol{x}}{1 + k_3|\boldsymbol{x}|^2}. \tag{7.31}$$

Depending on whether k_3 is positive or negative, barrel- and cushion-shaped distortions in the images of squares will be observed. Commercial TV lenses show a radial deviation of several image points (pixels) at the edge of the sensor. If the distortion is corrected with Eq. (7.31), the residual error is less than 0.06 image points [107].

This high degree of correction, together with the geometric stability of modern CCD sensors, accounts for subpixel accuracy in distance and area measurements without using expensive special lenses. Lenz [108] discusses further details which influence the geometrical accuracy of CCD sensors.

Distortions also occur if non-planar surfaces are projected onto the image plane. These distortions prevail in satellite and aerial imagery. Thus correction of geometric distortion in images is a basic topic in remote sensing and photogrammetry [151].

Accurate correction of the geometrical distortions requires shifting of image points by fractions of the distance between two image points. We will deal with this problem later in Section 10.6 after we have worked out the knowledge necessary to handle it properly.

7.5 Radiometry of Imaging

If is not sufficient to know only the geometry of imaging. Equally important is to consider how the irradiance at the image plane is related to the radiance of the imaged objects and which parameters of an optical system influence this relationship. For a discussion of the fundamentals of radiometry, especially all terms describing the properties of radiation, we refer to Section 6.3.

The path of radiation from a light source to the image plane involves a chain of processes (see Fig. 6.1). In this section, we concentrate on the observation path (compare Fig. 6.1), i. e., how the radiation emitted from the object to be imaged is collected by the imaging system.

7.5.1 Radiance Invariance

An optical system collects part of the radiation emitted by an object (Fig. 7.10). We assume that the object is a Lambertian radiator with the radiance L. The aperture of the optical system appears from the object to subtend a certain solid angle Ω. The projected circular aperture area is $\pi r^2 \cos\theta$ at a distance $(d+f)/\cos\theta$. Therefore, a flux

$$\Phi = LA\frac{\pi r^2 \cos^3\theta}{(d+f)^2} \tag{7.32}$$

enters the optical system. The radiation emitted from the projected area A is imaged onto the area A'. Therefore, the flux Φ must be divided by

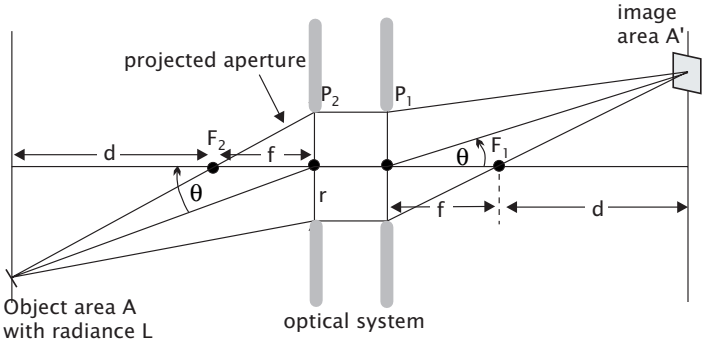

Figure 7.10: *An optical system receives a flux density that corresponds to the product of the radiance of the object and the solid angle subtended by the projected aperture as seen from the object. The flux emitted from the object area A is imaged onto the image area A′.*

the area A' in order to compute the image irradiance E'. The area ratio can be expressed as

$$A/A' = \frac{\cos^{-1}\theta\ (f+d)^2}{(f+d')^2}. \tag{7.33}$$

We further assume that the optical system has a transmittance t. This leads finally to the following object radiance/image irradiance relation:

$$E' = t\pi \left(\frac{r}{f+d'}\right)^2 \cos^4\theta\ L. \tag{7.34}$$

This fundamental relationship states that the image irradiance is proportional to the object radiance. This is the base for the linearity of optical imaging. The optical system is described by two simple terms: its (total) transmittance t and the ratio of the aperture radius to the distance of the image from the first principal point. For distant objects $d \gg f$, $d' \ll f$, Eq. (7.34) reduces to

$$E' = t\pi \frac{\cos^4\theta}{n_f^2} L \qquad d \gg f \tag{7.35}$$

using the f-number n_f. For real optical systems, equations Eqs. (7.34) and (7.35) are only an approximation. If part of the incident beam is cut off by additional apertures or limited lens diameters (*vignetting*), the fall-off is even steeper at high angles θ. On the other hand, a careful design of the position of the aperture can make the fall-off less steep than $\cos^4\theta$. As also the residual reflectivity of the lens surfaces depends on the angle of incidence, the true fall-off depends strongly on the design of

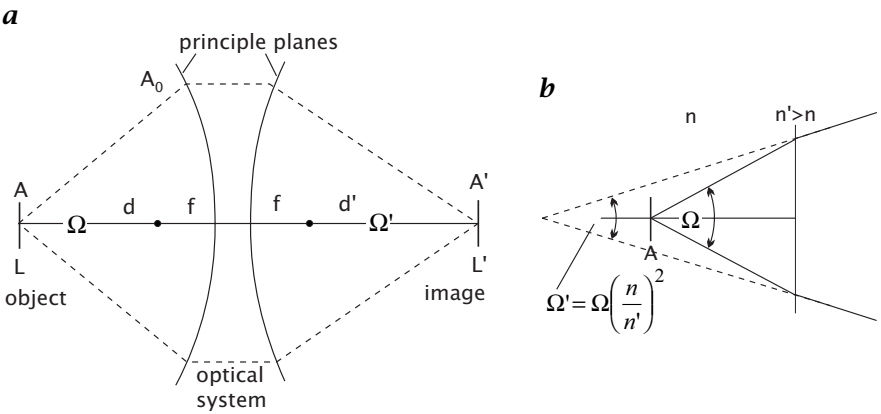

Figure 7.11: *Illustration of radiance invariance:* ***a*** *The product $A\Omega$ is the same in object and image space.* ***b*** *Change of solid angle, when a beam enters an optically denser medium.*

the optical system and is best determined experimentally by a suitable calibration setup.

The astonishing fact that the image irradiance is so simply related to the object radiance has its cause in a fundamental invariance. An image has a radiance just like a real object. It can be taken as a source of radiation by further optical elements. A fundamental theorem of radiometry now states that the radiance of an image is equal to the radiance of the object times the transmittance of the optical system.

The theorem can be proved using the assumption that the radiative flux Φ through an optical system is preserved except for absorption in the system leading to a transmittance less than one. The solid angles which the object and image subtend in the optical system are

$$\Omega = A_0/(d+f)^2 \quad \text{and} \quad \Omega' = A_0/(d'+f)^2, \tag{7.36}$$

where A_0 is the effective area of the aperture.

The flux emitted from an area A of the object is received by the area $A' = A(d'+f)^2/(d+f)^2$ in the image plane (Fig. 7.11a). Therefore, the radiances are

$$\begin{aligned} L &= \frac{\Phi}{\Omega A} = \frac{\Phi}{A_0 A}(d+f)^2 \\ L' &= \frac{t\Phi}{\Omega' A'} = \frac{t\Phi}{A_0 A}(d+f)^2, \end{aligned} \tag{7.37}$$

and the following invariance holds:

$$L' = tL \quad \text{for} \quad n' = n. \tag{7.38}$$

The radiance invariance of this form is only valid if the object and image are in media with the same refractive index ($n' = n$). If a beam with

radiance L enters a medium with a higher refractive index, the radiance increases as the rays are bent towards the optical axis (Fig. 7.11b). Thus, more generally the ratio of the radiance and the refractive index squared remains invariant:

$$L'/n'^2 = tL/n^2 \tag{7.39}$$

From the radiance invariance, we can immediately infer the irradiance on the image plane to be

$$E' = L'\pi \sin^2 \alpha' = tL\pi \sin^2 \alpha'. \tag{7.40}$$

This equation does not consider the fall-off with $\cos^4 \theta$ in Eq. (7.34) because we did not consider oblique principal rays. The term $\sin^2 \alpha$ corresponds to $r^2/(f+d')^2$ in Eq. (7.34). Radiance invariance considerably simplifies computation of image irradiance and the propagation of radiation through complex optical systems. Its fundamental importance can be compared to the principles in geometric optics that radiation propagates in such a way that the optical path nd (real path times the index of refraction) takes an extreme value.

7.6 Linear System Theory of Imaging†

In Section 4.2 we discussed linear shift-invariant filters (convolution operators) as one application of linear system theory. Imaging is another example that can be described with this powerful concept. Here we will discuss optical imaging in terms of the 2-D and 3-D point spread function (Section 7.6.1) and optical transfer function (Section 7.6.2).

7.6.1 Point Spread Function

Previously it was seen that a point in the 3-D object space is not imaged onto a point in the image space but onto a more or less extended area with varying intensities. Obviously, the function that describes the imaging of a point is an essential feature of the imaging system and is called the *point spread function*, abbreviated as *PSF*. We assume that the PSF is not dependent on position. Then optical imaging can be treated as a *linear shift-invariant system* (*LSI*) (Section 4.2).

If we know the PSF, we can calculate how any arbitrary 3-D object will be imaged. To perform this operation, we think of the object as decomposed into single points. Figure 7.12 illustrates this process. A point X' at the object plane is projected onto the image plane with an intensity distribution corresponding to the point spread function h. With $g'_i(\boldsymbol{x}')$ we denote the intensity values at the object plane $g'_o(\boldsymbol{X}')$ projected onto the image plane but without any defects through the imaging. Then the

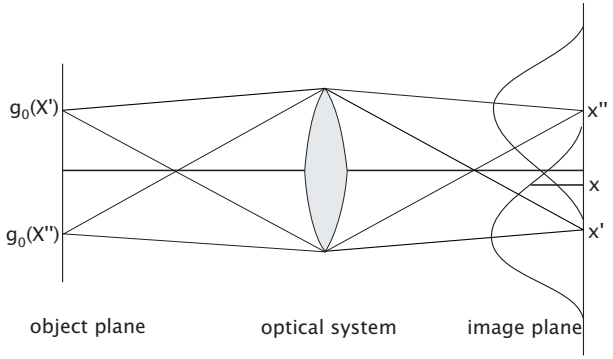

Figure 7.12: *Image formation by convolution with the point spread function $h(x)$. A point at X' in the object plane results in an intensity distribution with a maximum at the corresponding point x' on the image plane. At a point x on the image plane, the contributions from all points x', i. e., $g_i'(x')h(x - x')$, must be integrated.*

intensity of a point $\boldsymbol{x}$ at the image plane is computed by integrating the contributions from the point spread functions which have their maximums at x' (Fig. 7.12):

$$g_i(\boldsymbol{x}) = \int_{-\infty}^{\infty} g_i'(\boldsymbol{x}')h(\boldsymbol{x} - \boldsymbol{x}')\mathrm{d}^2x' = (g_i' * h)(\boldsymbol{x}). \tag{7.41}$$

The operation in Eq. (7.41) is known as a *convolution*. Convolutions play an essential role in image processing. Convolutions are not only involved in image formation but also in many image processing operations. In case of image formation, a convolution obviously "smears" an image and reduces the resolution.

This effect of convolutions can be most easily demonstrated with image structures that show periodic gray value variations. As long as the repetition length, the *wavelength*, of this structure is larger than the width of the PSF, it will suffer no significant changes. As the wavelength decreases, however, the amplitude of the gray value variations will start to decrease. Fine structures will finally be smeared out to such an extent that they are no longer visible. These considerations emphasize the important role of periodic structures and lead naturally to the introduction of the *Fourier transform* which decomposes an image into the periodic gray value variations it contains (Section 2.3).

Previous considerations showed that formation of a two-dimensional image on the image plane is described entirely by its PSF. In the following we will extend this concept to three dimensions and explicitly calculate the point spread function within the limit of geometric optics, i. e., with a perfect lens system and no diffraction. This approach is motivated

by the need to understand three-dimensional imaging, especially in microscopy, i. e., how a point in the 3-D object space is imaged not only onto a 2-D image plane but into a 3-D image space.

First, we consider how a fixed point in the object space is projected into the image space. From Fig. 7.8 we infer that the radius of the unsharpness disk is given by

$$\epsilon_i = \frac{r x_3}{d_i}. \tag{7.42}$$

The index i of ε indicates the image space. Then we replace the radius of the aperture r by the maximum angle under which the lens collects light from the point considered and obtain

$$\epsilon_i = \frac{d_o}{d_i} x_3 \tan\alpha. \tag{7.43}$$

This equation gives us the edge of the PSF in the image space. It is a double cone with the x_3 axis in the center. The tips of both the cones meet at the origin. Outside the two cones, the PSF is zero. Inside the cone, we can infer the intensity from the conservation of radiation energy. Since the radius of the cone increases linearly with the distance to the plane of focus, the intensity within the cone decreases quadratically. Thus the PSF $h_i(\boldsymbol{x})$ in the image space is given by

$$\begin{aligned} h_i(\boldsymbol{x}) &= \frac{I_0}{\pi(\frac{d_o}{d_i} x_3 \tan\alpha)^2} \Pi \frac{(x_1^2 + x_2^2)^{1/2}}{2\frac{d_o}{d_i} x_3 \tan\alpha} \\ &= \frac{I_0}{\pi(\frac{d_o}{d_i} z \tan\alpha)^2} \Pi \frac{r}{2\frac{d_o}{d_i} z \tan\alpha}, \end{aligned} \tag{7.44}$$

where I_0 is the light intensity collected by the lens from the point, and Π is the *box function*, which is defined as

$$\Pi(x) = \begin{cases} 1 & |x| \le 1/2 \\ 0 & \text{otherwise} \end{cases}. \tag{7.45}$$

The last expression in Eq. (7.44) is written in cylindrical coordinates (r, ϕ, z) to take into account the circular symmetry of the PSF with respect to the x_3 axis.

In a second step, we discuss what the PSF in the image space refers to in the object space, since we are interested in how the effects of the imaging are projected back into the object space. We have to consider both the lateral and axial magnification. First, the image, and thus also ε, are larger than the object by the factor d_i/d_o. Second, we must find the planes in object and image space corresponding to each other. This problem has already been solved in Section 7.4.2. Equation Eq. (7.23)

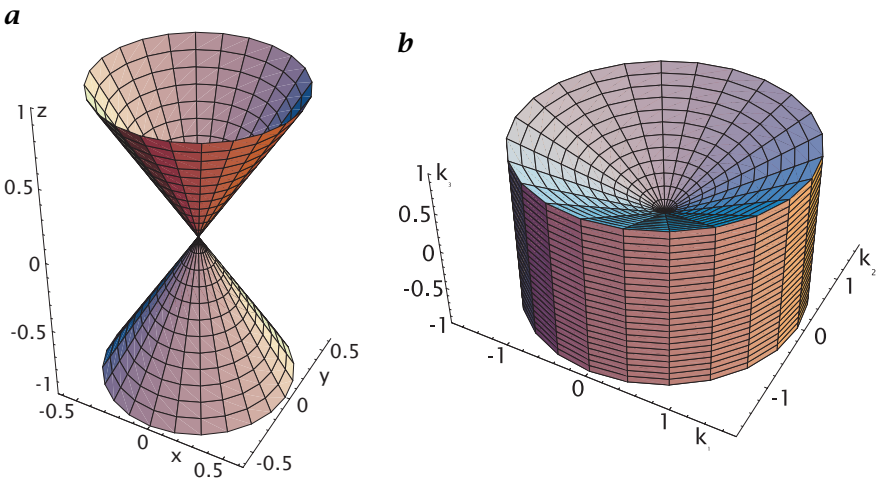

Figure 7.13: ***a*** *3-D PSF and* ***b*** *3-D OTF of optical imaging with a lens, back-projected into the object space. Lens aberrations and diffraction effects are neglected.*

relates the image to the camera coordinates. In effect, the back-projected radius of the unsharpness disk, ϵ_o, is given by

$$\epsilon_o = X_3 \tan\alpha, \tag{7.46}$$

and the PSF, back-projected into the object space, by

$$\begin{aligned} h_o(\boldsymbol{X}) &= \frac{I_0}{\pi (X_3 \tan\alpha)^2} \Pi \frac{(X_1^2 + X_2^2)^{1/2}}{2X_3 \tan\alpha} \\ &= \frac{I_0}{\pi (Z \tan\alpha)^2} \Pi \frac{R}{2Z \tan\alpha}. \end{aligned} \tag{7.47}$$

The double cone of the PSF, back-projected into the object space, shows the same opening angle as the lens (Fig. 7.13). In essence, $h_0(x)$ in Eq. (7.47) gives the effect of optical imaging disregarding geometric scaling.

7.6.2 Optical Transfer Function

Convolution with the PSF in the space domain is a quite complex operation. In Fourier space, however, it is performed as a multiplication of complex numbers. In particular, convolution of the 3-D object $g'_o(\boldsymbol{X})$ with the PSF $h_o(\boldsymbol{X})$ corresponds in Fourier space to a multiplication of the Fourier transformed object $\hat{g}'_o(\boldsymbol{k})$ with the Fourier transformed PSF, the *optical transfer function* or *OTF* $\hat{h}_o(\boldsymbol{k})$. In this section, we consider

the optical transfer function in the object space, i. e., we project the imaged object back into the object space. Then the image formation can be described by:

	Imaged object		Imaging		Object	(7.48)
Space domain	$g_o(\boldsymbol{X})$	=	$h_o(\boldsymbol{X})$	$*$	$g'_o(\boldsymbol{X})$	
Fourier domain	$\hat{g}_o(\boldsymbol{k})$	=	$\hat{h}_o(\boldsymbol{k})$	$\cdot$	$\hat{g}'_o(\boldsymbol{k})$.	

This correspondence means that we can describe optical imaging with either the point spread function or the optical transfer function. Both descriptions are complete. As with the PSF, the OTF has an illustrative meaning. As the Fourier transform decomposes an object into periodic structures, the OTF tells us how these periodic structures are changed by the optical imaging process. An OTF of 1 for a particular wavelength means that this periodic structure is not affected at all. If the OTF is 0, it disappears completely. For values between 0 and 1 it is attenuated correspondingly. Since the OTF is generally a complex number, not only the amplitude of a periodic structure can be changed but also its phase.

Direct calculation of the OTF is awkward.

Here several features of the Fourier transform are used, especially its linearity and separability, to decompose the PSF into suitable functions which can be transformed more easily. Two possibilities are demonstrated. They are also more generally instructive, since they illustrate some important features of the Fourier transform.

The first method for calculating the OTF decomposes the PSF into a bundle of δ lines intersecting at the origin of the coordinate system. They are equally distributed in the cross-section of the double cone. We can think of each δ line as being one light ray. Without further calculations, we know that this decomposition gives the correct quadratic decrease in the PSF, because the same number of δ lines intersect a quadratically increasing area. The Fourier transform of a δ line is a δ plane which is perpendicular to the line ($\succ$ R5). Thus the OTF is composed of a bundle of δ planes. They intersect the k_1k_2 plane at a line through the origin of the k space under an angle of at most α. As the Fourier transform preserves rotational symmetry, the OTF is also circular symmetric with respect to the k_3 axis. The OTF fills the whole Fourier space except for a double cone with an angle of $\pi/2 - \alpha$. In this sector the OTF is zero. The exact values of the OTF in the non-zero part are difficult to obtain with this decomposition method.

We will infer it with another approach, based on the separability of the Fourier transform. We think of the double cone as layers of disks with varying radii which increase with $|x_3|$. In the first step, we perform the Fourier transform only in the x_1x_2 plane. This transformation yields a function with two coordinates in the k space and one in the x space,

(k_1, k_2, x_3), respectively (q, φ, z) in cylinder coordinates. Since the PSF Eq. (7.47) depends only on r (rotational symmetry around the z axis), the two-dimensional Fourier transform corresponds to a one-dimensional *Hankel transform* of zero order [11]:

$$\begin{aligned} h(r,z) &= \frac{I_0}{\pi (z\tan\alpha)^2}\Pi\left(\frac{r}{2z\tan\alpha}\right) \\ \check{h}(q,z) &= I_0 \frac{\mathrm{J}_1(2\pi z q\tan\alpha)}{\pi z q\tan\alpha}. \end{aligned} \tag{7.49}$$

The Fourier transform of the disk thus results in a function that contains the *Bessel function* J_1 (≻ R5).

As a second step, we perform the missing one-dimensional Fourier transform in the z direction. Equation Eq. (7.49) shows that $\check{h}(q,z)$ is also a Bessel function in z. This time, however, the Fourier transform is one-dimensional. Thus we obtain not a disk function but a circle function (≻ R5):

$$\frac{\mathrm{J}_1(2\pi x)}{x} \quad \circ\!\!-\!\!\bullet \quad 2\left(1-k^2\right)^{1/2}\Pi\left(\frac{k}{2}\right). \tag{7.50}$$

If we finally apply the Fourier transform *scaling theorem* (≻ R4),

$$\begin{aligned} &\text{if} \quad f(x) \quad \circ\!\!-\!\!\bullet \quad \hat{f}(k), \\ &\text{then} \quad f(ax) \quad \circ\!\!-\!\!\bullet \quad \frac{1}{|a|}\hat{f}\left(\frac{k}{a}\right), \end{aligned} \tag{7.51}$$

we obtain

$$\hat{h}(q,k_3) = \frac{2I_0}{\pi|q\tan\alpha|}\left(1-\frac{k_3^2}{q^2\tan^2\alpha}\right)^{1/2}\Pi\left(\frac{k_3}{2q\tan\alpha}\right). \tag{7.52}$$

A large part of the OTF is zero. This means that spatial structures with the corresponding directions and wavelengths completely disappear. In particular, this is the case for all structures in the z direction, i. e., perpendicular to the image plane. Such structures get completely lost and cannot be reconstructed without additional knowledge.

We can only see 3-D structures if they also contain structures parallel to the image plane. For example, it is possible to resolve points or lines that lie above each other. We can explain this in the x space as well as in the k space. The PSF blurs the points and lines, but they can still be distinguished if they are not too close to each other.

Points or lines are extended objects in Fourier space, i. e., constants or planes. Such extended objects partly coincide with the non-zero parts of the OTF and thus will not vanish entirely. Periodic structures up to an angle of α to the k_1k_2 plane, which just corresponds to the opening angle of the lens, are not eliminated by the OTF. Intuitively, we can say

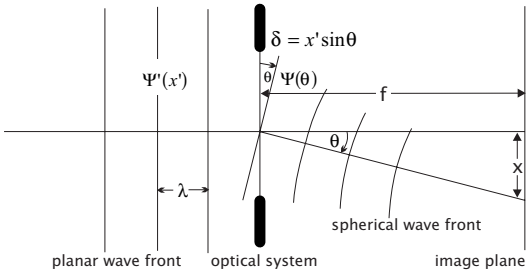

Figure 7.14: *Diffraction of a planar wave front at the aperture stop of an optical system. At the aperture stop we think of the planar wave as being decomposed into spherical wave fronts that show a difference of $\delta = x' \sin\theta$ depending on their direction θ and position x'.*

that we are able to recognize all 3-D structures that we can actually look into. All we need is at least one ray which is perpendicular to the wave number of the structure and, thus, run in the direction of constant gray values.

7.6.3 Diffraction-Limited Optical Systems‡

Light is electromagnetic radiation and as such subject to wave-related phenomena. When a parallel bundle of light enters an optical system, it cannot be focused to a point even if all aberrations have been eliminated. Diffraction at the aperture of the optical system blurs the spot at the focus to a size of at least the order of the wavelength of the light. An optical system for which the aberrations have been suppressed to such an extent that it is significantly lower than the effects of diffraction is called *diffraction-limited*.

A rigorous treatment of diffraction according to Maxwell's equations is mathematically quite involved ([34, Chapters 9 and 10] and [77, Chapter 3]). The diffraction of a planar wave at the aperture of lenses can be treated in a simple approximation known as *Fraunhofer diffraction*. It leads to a fundamental relation.

We assume that the aperture of the optical system is pierced by a planar wave front (Fig. 7.14). At the aperture plane, we apply Huygens' principle which states that each point of the wave front can be taken as the origin of a new in-phase spherical wave. In particular, we can add up these waves to new planar wave fronts leaving the aperture plane at an angle of θ. The inclination causes a path difference which results in a position-dependent phase shift. The path difference in the aperture plane is $\delta = x' \sin\theta$, where x' is the position in the aperture plane. Thus, the phase difference is

$$\Delta\varphi = 2\pi\delta/\lambda = 2\pi x' \sin\theta/\lambda, \tag{7.53}$$

where λ is the wavelength of the wave front. We further assume that $\psi'(x')$ is the amplitude distribution of the wave front at the aperture plane. In case of a simple aperture stop, $\psi'(x')$ is a simple box function, but we want to treat

the more general case of a varying amplitude of the wave front or any type of aperture functions. Then, the planar wave front leaving the aperture under an angle θ is given by the integral

$$\psi(\theta) = \int_{-\infty}^{\infty} \psi'(x') \exp\left(\frac{2\pi i x' \sin\theta}{\lambda}\right) \mathrm{d}x'. \tag{7.54}$$

This equation describes the diffraction pattern at infinity as a function of the angle θ. We have not yet included the effect of the optical system. All that an ideal optical system does, is to bend the planar wave front at the aperture into a spherical wave that converges at the image plane into a point. This point is given by the relation $x = f \tan\theta \approx f \sin\theta$. With this relation Eq. (7.54) becomes

$$\psi(x) = \int_{-\infty}^{\infty} \psi'(x') \exp\left(\frac{2\pi i x' x}{f\lambda}\right) \mathrm{d}x'. \tag{7.55}$$

This integral can easily be extended to two dimensions by replacing $x'x$ with the scalar product of the 2-D vectors $\boldsymbol{x}'$ and $\boldsymbol{x}$:

$$\psi(\boldsymbol{x}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \psi'(\boldsymbol{x}') \exp\left(2\pi i \frac{\boldsymbol{x}'^T\boldsymbol{x}}{f\lambda}\right) \mathrm{d}^2x'. \tag{7.56}$$

These equation means that the amplitude distribution $\psi(x)$ at the focal plane is simply the 2-D Fourier transform of the amplitude function $\psi'(x')$ at the aperture plane.

For a *circular aperture*, the amplitude distribution is given by

$$\psi'(\boldsymbol{x}') = \Pi\left(\frac{|\boldsymbol{x}'|}{2r}\right), \tag{7.57}$$

where r is the radius of the aperture. The Fourier transform of Eq. (7.57) is given by the Bessel function of first order (≻ R4):

$$\psi(\boldsymbol{x}) = \psi_0 \frac{I_1(2\pi x r/f\lambda)}{\pi x r/f\lambda}. \tag{7.58}$$

The irradiance E on the image plane is given by the square of the amplitude:

$$E(\boldsymbol{x}) = |\psi(\boldsymbol{x})|^2 = \psi_0^2 \left(\frac{I_1(2\pi x r/f\lambda)}{\pi x r/f\lambda}\right)^2. \tag{7.59}$$

The diffraction pattern has a central spot that contains 83.9 % of the energy and encircling rings with decreasing intensity (Fig. 7.15a). The distance from the center of the disk to the first dark ring is

$$\Delta x = 0.61 \cdot \frac{f}{r}\lambda = 1.22\lambda n_f. \tag{7.60}$$

At this distance, two points can clearly be separated (Fig. 7.15b). This is the *Rayleigh criterion* for resolution of an optical system. The resolution of an

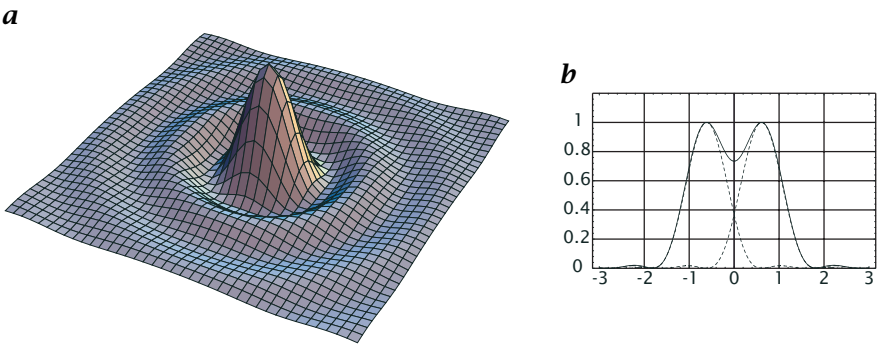

Figure 7.15: ***a*** *Irradiance $E(\boldsymbol{x})$ of the diffraction pattern ("Airy disk") at the focal plane of an optical system with a uniformly illuminated circular aperture according to Eq. (7.59).* ***b*** *Illustration of the resolution of the image of two points at a distance $x/(n_f\lambda) = 1.22$.*

optical system can be interpreted in terms of the angular resolution of the incoming planar wave and the spatial resolution at the image plane. Taking the Rayleigh criterion Eq. (7.60), the angular resolution $\Delta\theta_0 = \Delta x/f$ is given as

$$\Delta\theta_0 = 0.61\frac{\lambda}{r}. \tag{7.61}$$

Thus, the angular resolution does not depend at all on the focal length but only the aperture of the optical system in relation to the wavelength of the electromagnetic radiation.

In contrast to the angular resolution, the spatial resolution Δx at the image plane, depends according to Eq. (7.60) only on the relation of the radius of the lens aperture to the distance f of the image of the object from the principal point. Instead of the f-number we can use in Eq. (7.60) the *numerical aperture* which is defined as

$$n_a = n\sin\theta_0 = \frac{2n}{n_f}. \tag{7.62}$$

We assume now that the image-sided index of refraction n may be different from 1. Here θ_0 is the opening angle of the light cone passing from the center of the image plane through the lens aperture. Then

$$\Delta x = 0.61\frac{\lambda}{n_a'}. \tag{7.63}$$

Therefore, the absolute resolution at the image plane does not at all depend again on the focal length of the system but only the numerical aperture of the image cone.

As the light way can be reversed, the same arguments apply for the object plane. The spatial resolution at the object plane depends only on the numerical aperture of the object cone, i. e., the opening angle of the cone entering the lens aperture:

$$\Delta X = 0.61\frac{\lambda}{n_a}. \tag{7.64}$$

These simple relations are helpful to evaluate the performance of optical systems. Since the maximum numerical aperture of optical systems is about one, no smaller structures than about half the wavelength can be resolved.

7.7 Further Readings‡

In this chapter, only the basic principles of imaging techniques are discussed. A more detailed discussion can be found in Jähne [81] or Richards [150]. The geometrical aspects of imaging are also of importance for computer graphics and are therefore treated in detail in standard textbooks on computer graphics, e. g. Watt [194] or Foley et al. [46]. More details about optical engineering can be found in the following textbooks: Iizuka [77] (especially about Fourier optics) and Smith [174]. Riedl [153] focuses on the design of infrared optics. In this chapter, the importance of linear system theory has been stressed for the description of an optical system. Linear system theory has widespread applications throughout science and engineering, see, e. g., Close and Frederick [21] or Dorf and Bishop [32].

8 3-D Imaging

8.1 Basics

In this chapter we discuss various imaging techniques that can retrieve the depth coordinate which is lost by the projection of the object onto an image plane. These techniques fall into two categories. They can either retrieve only the depth of a surface in 3-D space or allow for a full reconstruction of volumetric objects. Often *depth imaging* and *volumetric imaging* are both called *3-D imaging*. This causes a lot of confusion.

Even more confusing is the wide variety of both depth and volumetric imaging techniques. Therefore this chapter will not detail all available techniques. It rather focuses on the basic principles. Surprisingly or not, there are only a few principles on which the wide variety of 3-D imaging techniques is based. If you know them, it is easy to understand how they work and what accuracy you can expect.

We start with the discussion of the basic limitation of projective imaging for 3-D vision in Section 8.1.1 and then give a brief summary of the basic principles of depth imaging (Section 8.1.2) and volumetric imaging (Section 8.1.3). Then one section is devoted to each of the basic principles of 3-D imaging: depth from triangulation (Section 8.2), depth from time-of-flight (Section 8.3), depth from phase (interferometry) (Section 8.4), shape from shading and photogrammetric stereo (Section 8.5), and tomography (Section 8.6).

8.1.1 Basic Limitation of Projective Imaging

As we have discussed in detail in Sections 7.6.1 and 7.6.2, a projective optical system is a linear shift-invariant system that can be described by a point spread function (PSF) and optical transfer function (OTF).

The 3-D OTF for geometrical optics shows the limitations of a projective imaging system best (see Section 7.6.2):

$$\hat{h}(q,k_3) = \frac{2I_0}{\pi|q\tan\alpha|}\left(1-\frac{k_3^2}{q^2\tan^2\alpha}\right)^{1/2}\Pi\left(\frac{k_3}{2q\tan\alpha}\right). \tag{8.1}$$

The symbols q and k_3 denote the radial and axial components of the wave number vector, respectively. Two severe limitations of 3-D imaging immediately follow from the shape of the 3-D OTF.

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2
Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

Complete Loss in Wide Wave Number Range. As shown in Fig. 7.13b, the 3-D OTF is rotationally symmetric around the k_3 axis (z direction) and nonzero only inside an angle cone of $\pm\alpha$ around the xy plane. Structures with a wide range of wave numbers especially around the z axis are completely lost. We can "see" only structures in those directions from which the optics collect rays.

Loss of Contrast at High Wave Numbers. According to Eq. (8.1), the OTF is inversely proportional to the radial wave number q. Consequently, the contrast of a periodic structure is attenuated in proportion to its wave number. As this property of the OTF is valid for all optical imaging — including the human visual system — the question arises why can we see fine structures at all.

The answer lies in a closer examination of the geometric structure of the objects observed. Most objects in the natural environment are opaque. Thus, we see only the surfaces, i. e., we do not observe real 3-D objects but only 2-D surface structures. If we image a 2-D surface onto a 2-D image plane, the 3-D PSF also reduces to a 2-D function. Mathematically, this means a multiplication of the PSF with a δ plane parallel to the observed surface. Consequently, the 2-D PSF is now given by the unsharpness disk corresponding to the distance of the surface from the lens. The restriction to 2-D surfaces thus preserves the intensity of all structures with wavelengths larger than the disk. We can see them with the same contrast.

We arrive at the same conclusion in Fourier space. Multiplication of the 3-D PSF with a δ plane in the x space corresponds to a convolution of the 3-D OTF with a δ line along the optical axis, i. e., an integration in the corresponding direction. If we integrate the 3-D OTF along the k coordinate, we actually get a constant independent of the radial wave number q:

$$\frac{2I_0}{\pi}\int_{-q\tan\alpha}^{q\tan\alpha}\frac{1}{|q\tan\alpha|}\left[1-\left(\frac{z'}{q\tan\alpha}\right)^2\right]^{1/2}\mathrm{d}z' = I_0. \tag{8.2}$$

To solve the integral, we substitute $z'' = z'/(q\tan\alpha)$ which yields an integral over a unit semicircle.

In conclusion, there is a significant difference between surface imaging (and thus *depth imaging*) and *volumetric imaging*. The OTF for surface structures is independent of the wave number. However, for volumetric structures, we still have the problem of the decrease of the OTF with the radial wave number. When observing such structures by eye or with a camera, we will not be able to observe fine details. Projective imaging systems are not designed to image true 3-D objects. Consequently, volumetric imaging requires different techniques.

8.1.2 Basic Principles of Depth Imaging

Depth imaging of a single opaque surface requires one additional piece of information besides the brightness at each pixel of the image in order to produce a depth image or range image. We can distinguish three basic principles of depth imaging known as *depth from paradigms*:

Depth from Triangulation. If we observe an object from two different points of view separated by a base line b, the object will be seen under a different angle to the base line from both positions. This technique is known as *triangulation* and constitutes one of the basic techniques in *geodesy* and *cartography*.

The triangulation technique is at the heart of a surprisingly wide variety of techniques. At first glance these techniques appear so different that it is difficult to believe that they are based on the same principle.

Depth from Time-of-Flight. This is another straightforward principle of distance measurement. A signal is sent out, propagates with a characteristic speed to the object, is reflected and travels back to the camera. The travel time is directly proportional to the sum of the distances between the sender and the object and the object and the receiver.

Depth from Phase: Interferometry.

Interferometry can be regarded as a special form of time-of-flight distance measurement. This technique measures distances as multiples of the wavelength of the radiation by measuring not only the amplitude (energy) of the radiation but also its phase. Phase measurements are possible by superimposition of coherent radiation (Section 6.2.3) leading to high intensities when the two superimposing wave fronts are in phase (constructive interference) and to low intensities when they show a phase shift of 180° (π, destructive interference). Light has wavelengths between 400 and 700 nm (Section 6.2.1 and Fig. 6.2). Consequently interferometric distance measurements with light resolve distances in the nanometer range (10^{-9} m) — a fraction of the wavelength.

Shape from Shading. The shape of surfaces can also be determined from the local orientation of the surface elements. This is expressed mathematically by the surface normal. Then, of course, the absolute depth of surface is lost, but the depth profile can be computed by integrating the surface inclination. The surface normal can be inferred from the shading because the radiance of a surface depends on the angle of incidence of the illumination source.

8.1.3 Basic Principles of Volumetric Imaging

Any depth from technique that can measure multiple depths simultaneously is also useful for volumetric imaging. The capability to measure multiple depths is thus another important characteristic of a depth imaging technique. In addition to the depth imaging techniques, there are two new basic principles for volumetric images:

Illumination Slicing. In projective imaging, we do not know from which depth the irradiance collected at the image plane originates. It could be from any position of the projection ray (see Section 7.3.1 and Fig. 7.3). However, the illumination can be arranged in such a way that only a certain depth range receives light. Then we know from which depth the irradiance at the image plane originates. When we scan the illumination depth, a volumetric image can be taken.

Depth from Multiple Projections: Tomography. A single projection contains only partial information from a volumetric object. The question therefore is, whether it is possible to take multiple projections from different directions and to combine the different pieces of partial information to a complete 3-D image. Such *depth from multiple projections* techniques are known as *tomography*.

8.1.4 Characterization of 3-D Imaging Techniques

Depth imaging is characterized by two basic quantities, the *depth resolution* σ_z and the *depth range* Δz. The depth resolution denotes the statistical error of the depth measurement and thus the minimal resolvable depth difference. Note that the systematic error of the depth measurement can be much larger (see discussion in Section 3.1). How the resolution depends on the distance z is an important characteristic of a depth imaging technique. It makes a big difference, for example, whether the resolution is uniform, i. e., independent of the depth, or decreasing with the distance z.

The depth range Δz is the difference between the minimum and maximum depth that can be measured by a depth imaging technique. Consequently, the ratio of the depth range and depth resolution, $\Delta z/\sigma_z$, denotes the *dynamic range* of depth imaging.

8.2 Depth from Triangulation

Looking at the same object from different points of view separated by a base vector $\boldsymbol{b}$ results in different viewing angles. In one way or the other, this difference in viewing angle results in a shift on the image

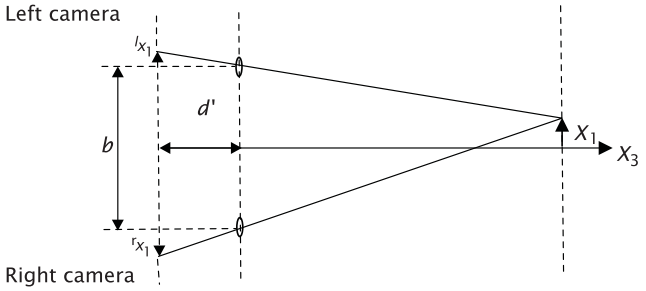

Figure 8.1: *A stereo camera setup.*

plane, known as *disparity*, from which the depth of the object can be inferred.

Triangulation-based depth measurements include a wide variety of different techniques that — at first glance — have not much in common, but are still based on the same principle. In this section we will discuss stereoscopy (Section 8.2.1), active triangulation, where one of the two cameras is replaced by a light source (Section 8.2.2), depth from focus (Section 8.2.3), and confocal microscopy (Section 8.2.4). In the section about stereoscopy, we also discuss the basic geometry of triangulation.

8.2.1 Stereoscopy

Observation of a scene from two different points of view allows the distance of objects to be determined. A setup with two imaging sensors is called a *stereo system*. Many biological visual systems perform depth perception in this way. Figure 8.1 illustrates how depth can be determined from a stereo camera setup. Two cameras are placed close to each other with parallel optical axes. The distance vector $\boldsymbol{b}$ between the two optical axes is called the *stereoscopic basis*.

An object will be projected onto different positions of the image plane because it is viewed from slightly different angles. The difference in the position is denoted as the *disparity* or *parallax*, p. It is easily calculated from Fig. 8.1:

$$p = {}^r x_1 - {}^l x_1 = d' \frac{X_1 + b/2}{X_3} - d' \frac{X_1 - b/2}{X_3} = b \frac{d'}{X_3}. \tag{8.3}$$

The parallax is inversely proportional to the distance X_3 of the object (zero for an object at infinity) and is directly proportional to the stereoscopic basis and the focal length of the cameras ($d' \approx f$ for distant objects). Thus the distance estimate becomes more difficult with increasing distance. This can be seen more clearly by using the law of

error propagation (Section 3.3.3) to compute the error of X_3 from :

$$X_3 = \frac{bd'}{p} \quad \rightsquigarrow \quad \sigma_{X_3} = \frac{bd'}{p^2}\sigma_p = \frac{X_3^2}{bd'}\sigma_p. \tag{8.4}$$

Therefore, the absolute sensitivity for a depth estimate decreases with the distance squared. As an example, we take a stereo system with a stereoscopic basis of 200 mm and lenses with a focal length of 100 mm. Then, at a distance of 10 m the change in parallax is about 200 μm/m (about 20 pixel/m), while it is only 2 μm/m (0.2 pixel/m) at a distance of 100 m.

Parallax is a vector quantity and parallel to the stereoscopic basis $\boldsymbol{b}$. This has the advantage that if the two cameras are exactly oriented we know the direction of the parallax beforehand. On the other hand, we cannot calculate the parallax in all cases. If an image sector does not show gray value changes in the direction of the stereo basis, then we cannot determine the parallax. This problem is a special case of the so-called *aperture problem* which occurs also in motion determination and will be discussed in detail in Section 14.2.2.

The depth information contained in stereo images can be perceived directly with a number of different methods. First, the left and right stereo image can be represented in one image, if one is shown in red and the other in green. The viewer uses spectacles with a red filter for the right and a green filter for the left eye. In this way, the right eye observes only the green and the left eye only the red image. This method — called the *anaglyph method* — has the disadvantage that no color images can be used. However, this method needs no special hardware and can be projected, shown on any RGB monitor, or printed out with standard printers.

Vertical stereoscopy also allows for the viewing of color stereo images [102]. The two component images are arranged one over the other. When viewed with prism spectacles that refract the upper image to the right eye and the lower image to the left eye, both images fuse into a 3-D image.

Other stereoscopic imagers use dedicated hardware. A common principle is to show the left and right stereo image in fast alternation on a monitor and switch the polarization direction of the screen synchronously. The viewer wears polarizing spectacles that filter the correct images out for the left and right eye.

However, the anaglyph method has the largest potential for most applications, as it can be used with almost any image processing workstation, the only additional piece of hardware needed being red/green spectacles. A stimulating overview of scientific and technical applications of stereo images is given by Lorenz [115].

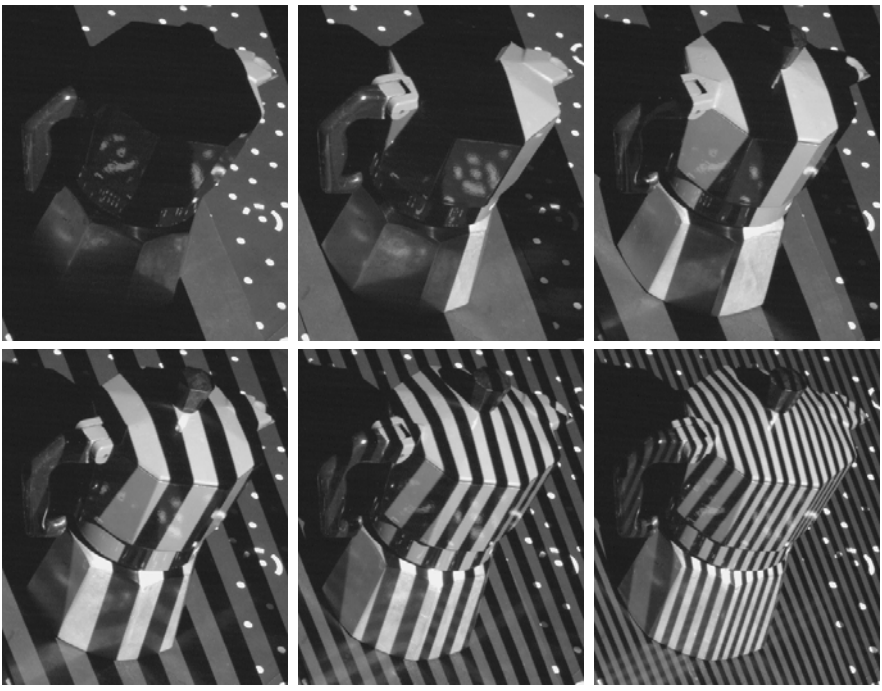

Figure 8.2: *Active triangulation by projection of a series of fringe patterns with different wavelengths for binary coding of the horizontal position; from Wiora [199].*

8.2.2 Depth from Active Triangulation

Instead of a stereo camera setup, one camera can be replaced by a light source. For a depth recovery it is then necessary to identify at each pixel from which direction the illumination is coming. This knowledge is equivalent to knowledge of the disparity. Thus an active triangulation technique shares all basic features with the stereo system that we discussed in the previous section.

Sophisticated techniques have been developed in recent years to code the light rays in a unique way. Most commonly, light projectors are used that project fringe patterns with stripes perpendicular to the triangulation base line onto the scene. A single pattern is not sufficient to identify the position of the pattern on the image plane in a unique way, but with a sequence of fringe patterns with different wavelengths, each horizontal position at the image plane of the light projector can be identified by a unique sequence of dark and bright stripes. A partial series of six such patterns is shown in Fig. 8.2.

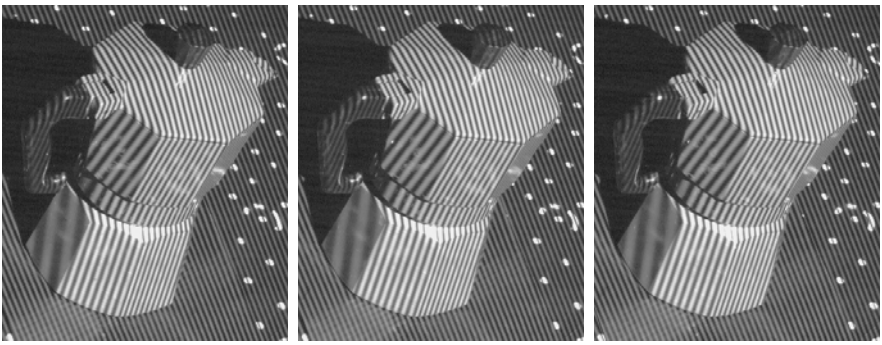

Figure 8.3: *Active triangulation by phase-shifted fringe patterns with the same wavelength. Three of four patterns are shown with phase shifts of 0, 90, and 180 degrees; from Wiora [199].*

Such a sequence of fringe patterns also has the advantage that — within the limits of the dynamic range of the camera — the detection of the fringe patterns becomes independent of the reflection coefficient of the object and the distance-dependent irradiance of the light projector. The occlusion problem that is evident from the shadow behind the espresso machine in Fig. 8.2 remains.

The binary coding by a sequence of fringe patterns no longer works for fine fringe patterns. For high-resolution position determination, as shown in Fig. 8.3, phase-shifted patterns of the same wavelength work much better and result in a subpixel-accurate position at the image plane of the light projector. Because the phase shift is only unique within a wavelength of the fringe pattern, in practice a hybrid code is often used that determines the coarse position by binary coding and the fine position by phase shifting.

8.2.3 Depth from Focus

The limited *depth of field* of a real optical system (Section 7.4.3) is another technique for depth estimation. An object is only imaged without blurring if it is within the depth of field. At first glance, this does not look like a depth from triangulation technique. However, it has exactly the same geometry as the triangulation technique. The only difference is that instead of two, multiple rays are involved and the radius of the blurred disk replaces the disparity. The triangulation base corresponds to the diameter of the optics. Thus depth from focus techniques share all the basic properties of a triangulation technique. For given optics, the resolution decreases with the square of the distance (compare Eq. (8.4) with Eq. (7.28)).

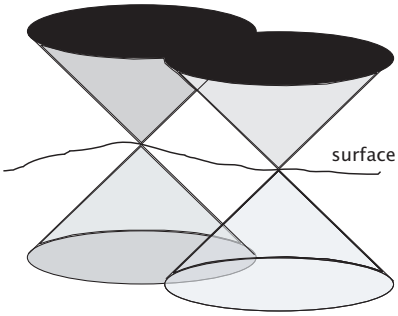

Figure 8.4: *Superposition of the point spread function of two neighboring points on a surface.*

The discussion on the limitations of projective imaging in Section 8.1.1 showed that the depth from focus technique does not work for volumetric imaging, because most structures, especially those in the direction of the optical axis, vanish. Depth from focus is, however, a very useful and simple technique for depth determination for opaque surfaces.

Steurer et al. [177] developed a simple method to reconstruct a *depth map* from a light microscopic focus series. A depth map is a two-dimensional function that gives the depth of an object point d — relative to a reference plane — as a function of the image coordinates $[x, y]^T$.

With the given restrictions, only one depth value for each image point needs to be found. We can make use of the fact that the 3-D point spread function of optical imaging discussed in detail in Section 7.6.1 has a distinct maximum in the focal plane because the intensity falls off with the square of the distance from the focal plane. This means that at all points where we get distinct image points such as edges, lines, or local extremes, we will also obtain an extreme in the gray value on the focal plane. Figure 8.4 illustrates that the point spread functions of neighboring image points only marginally influence each other close to the focal plane.

Steurer's method makes use of the fact that a distinct maximum of the point spread function exists in the focal plane. His algorithm includes the following four steps:

1. Take a focus series with constant depth steps.
2. Apply a suitable filter such as the *variance operator* (Section 15.2.2) to emphasize small structures. The highpass-filtered images are segmented to obtain a mask for the regions with significant gray value changes.
3. In the masked regions, search for the maximum magnitude of the difference in all the images of the focus series. The image in which the maximum occurs gives a depth value for the depth map. By in-

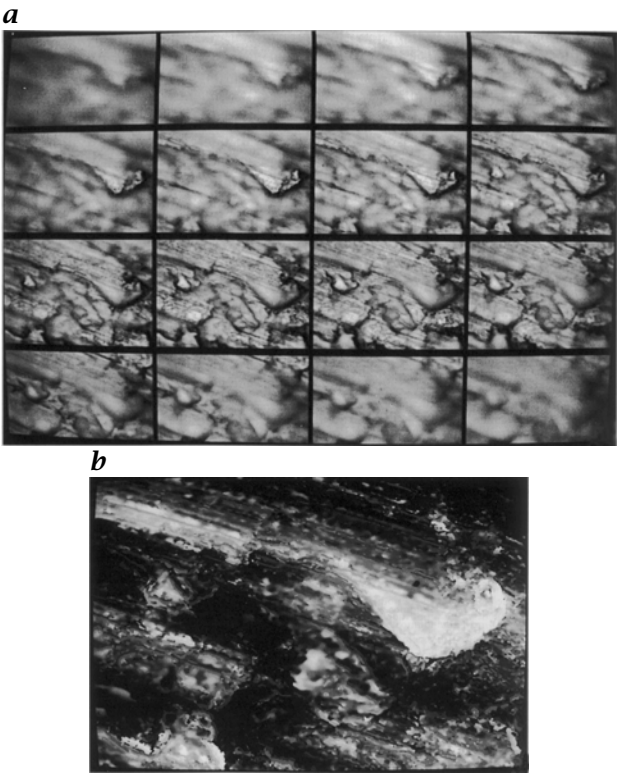

Figure 8.5: ***a*** *Focus series with 16 images of a metallic surface taken with depth distances of 2 µm; the focal plane becomes deeper from left to right and from top to bottom.* ***b*** *Depth map computed from the focus series. Depth is coded by intensity. Objects closer to the observer are shown brighter. From Steurer et al. [177].*

terpolation of the values the depth position of the maximum can be determined more exactly than with the depth resolution of the image series [163].

4. As the depth map will not be dense, interpolation is required. Steurer used a region-growing method followed by an adaptive lowpass filtering which is applied only to the interpolated regions in order not to corrupt the directly computed depth values. However, other valid techniques, such as normalized convolution (Section 11.7.2) or any of the techniques described in Section 17.3, are acceptable.

This method was successfully used to determine the surface structure of worked metal pieces. Figure 8.5 shows that good results were achieved. A filing can be seen that projects from the surface. Moreover, the surface shows clear traces of the grinding process.

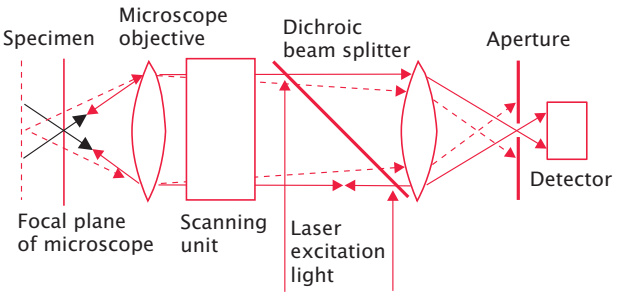

Figure 8.6: *Principle of confocal laser scanning microscopy.*

This technique works only if the surface shows fine details. If this is not the case, the confocal illumination technique of Scheuermann et al. [163] can be applied that projects statistical patterns into the focal plane (compare Section 1.2.2 and Fig. 1.3).

8.2.4 Confocal Microscopy

Volumetric microscopic imaging is of utmost importance for material and life sciences. Therefore the question arises, whether it is possible to change the image formation process — and thus the point spread function — so that the optical transfer function no longer vanishes, especially in the z direction.

The answer to this question is *confocal laser scanning microscopy*. Its basic principle is to illuminate only the points in the focal plane. This is achieved by scanning a laser beam over the image plane that is focused by the optics of the microscope onto the focal plane (Fig. 8.6). As the same optics are used for imaging and illumination, the intensity distribution in the object space is given approximately by the point spread function of the microscope. (Slight differences occur as the laser light is coherent.) Only a thin slice close to the focal plane receives a strong illumination. Outside this slice, the illumination falls off with the distance squared from the focal plane. In this way contributions from defocused objects outside the focal plane are strongly suppressed and the distortions decrease. However, can we achieve a completely distortion-free reconstruction? We will use two independent trains of thought to answer this question.

Let us first imagine a periodic structure in the z direction. In conventional microscopy, this structure is lost because all depths are illuminated with equal radiance. In confocal microscopy, however, we can still observe a periodic variation in the z direction because of the strong decrease of the illumination intensity provided that the wavelength in the z direction is not too small.

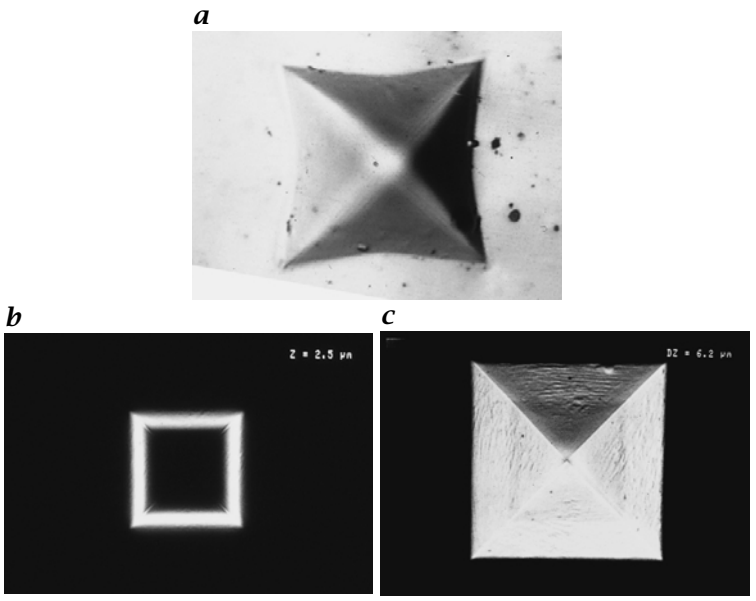

Figure 8.7: *Demonstration of confocal laser scanning microscopy (CLSM).* ***a*** *A square pyramid-shaped crystal imaged with standard microscopy focused on the base of the pyramid.* ***b*** *Similar object imaged with CLSM: only a narrow height contour range, 2.5 µm above the base of the square pyramid, is visible.* ***c*** *Image composed of a 6.2 µm depth range scan of CLSM images. Images courtesy of Carl Zeiss Jena GmbH, Germany.*

The same fact can be illustrated using the PSF. The PSF of confocal microscopy is given as the product of spatial intensity distribution and the PSF of the optical imaging. As both functions fall off with z^{-2}, the PSF of the confocal microscope falls off with z^{-4}. This much sharper localization of the PSF in the z direction results in a nonzero OTF in the z direction up to the z resolution limit.

The superior 3-D imaging of confocal laser scanning microscopy is demonstrated in Fig. 8.7. An image taken with standard microscopy shows a crystal in the shape of a square pyramid which is sharp only at the base of the pyramid (Fig. 8.7a). Towards the top of the pyramid, the edges become more blurred. In contrast, a single image taken with a confocal laser scanning microscopy images only a narrow height range at all (Fig. 8.7b). An image composed of a 6.2 μm depth scan by adding up all images shows a sharp image for the whole depth range (Fig. 8.7c). Many fine details can be observed that are not visible in the image taken with the conventional microscope. The laser scanning microscope has found widespread application in medical and biological sciences and materials research.

8.3 Depth from Time-of-Flight

Time-of-flight techniques measure the delay caused by the time for a signal to travel a certain distance. If the signal is sent out from the position of the camera, it has to travel twice the distance between the camera and the object reflecting the signal. Therefore the delay τ is given by

$$\tau = \frac{2z}{c}, \tag{8.5}$$

where c is the travel speed of the signal. From Eq. (8.5) it is evident that the statistical error of the depth measurement is independent of the distance to the object. It only depends on the accuracy of the delay measurement:

$$z = \frac{c\tau}{2} \quad \rightsquigarrow \quad \sigma_z = \frac{c}{2}\sigma_\tau. \tag{8.6}$$

This is a significant advantage over triangulation techniques (Eq. (8.4)).

With time-of-flight techniques one immediately thinks of *pulse modulation*, i. e., measuring the time of flight by the delay between sending and receiving a short pulse. The maximum measurable distance depends on the frequency with which the pulses are sent to the object. With electromagnetic waves, delay measurements are very demanding. Because the light speed c is $3 \cdot 10^8$ m/s, the delay is only 6.7 ns per meter.

Pulse modulation is only one of many techniques to modulate the signal for time of flight measurements. Another powerful technique is the *continuous-wave modulation* (CW modulation). With this technique the signal is modulated periodically and the delay is measured as a phase shift between the outgoing and ingoing signal:

$$z = \frac{c}{2\omega}\phi \quad \rightsquigarrow \quad \sigma_z = \frac{c}{2\omega}\sigma_\phi, \tag{8.7}$$

where ω is the circular frequency of the modulation. The depth range is given by the fact that the phase can be measured uniquely only in a range of $\pm\pi$:

$$\Delta z = \frac{\pi c}{\omega} = \frac{cT}{2}. \tag{8.8}$$

One of the most significant disadvantages of periodic modulation is thus the limited depth range. This problem is overcome by *pseudo-noise modulation* where the signal amplitude is randomly modulated. This technique combines the high resolution of CW modulation with the large distance range of pulse modulation.

8.4 Depth from Phase: Interferometry

Interferometry can be regarded as a special case of continuous-wave modulation. The modulation is given directly by the frequency of the electromagnetic radiation. It is still useful to regard interferometry as a

special class of range measurement technique because coherent radiation (Section 6.2.3) is required. Because of the high frequencies of light, the phases of the outgoing and incoming radiation cannot be measured directly but only by the amplitude variation caused by the coherent optical superimposition of the outgoing and incoming light.

The depth error and depth range for interferometric range measurements is simply given by Eqs. (8.7) and (8.8) and the relations $c = \nu\lambda$ and $\omega = 2\pi\nu = ck$ (Section 6.2.1):

$$z = \frac{1}{2k}\phi = \frac{\lambda}{4\pi}\phi, \quad \sigma_z = \frac{\lambda}{4\pi}\sigma_\phi, \quad \Delta z = \frac{\lambda}{2}. \tag{8.9}$$

Because of the small wavelength of light (0.4–0.7 μm), interferometric measurements are extremely sensitive. The limited depth range of only half a wavelength can be overcome by *multiwavelength interferometry*

A second class of interferometric range measuring techniques is possible with radiation that shows a coherence length of only a few wavelengths. Then interference patterns occur only for a short distance of a few wavelengths and can thus be taken as a depth measurement in a scanning system. This type of interferometry is known as *white-light interferometry* or *coherency radar*.

8.5 Shape from Shading†

Shape from shading techniques do not infer the depth but the normal of surfaces and thus form an entirely new class of surface reconstruction techniques. It is obvious that shape from shading techniques cannot infer absolute distances.

8.5.1 Shape from Shading for Lambertian Surfaces

We first apply this technique for diffuse reflecting opaque objects. For the sake of simplicity, we assume that the surface of a Lambertian object is illuminated by parallel light. The radiance L of a Lambertian surface (Section 6.4.3) does not depend on the viewing angle and is given by:

$$L = \frac{\rho(\lambda)}{\pi} E \cos\gamma, \tag{8.10}$$

where E is the irradiance and γ the angle between the surface normal and the illumination direction. The relation between the surface normal and the incident and exitant radiation can most easily be understood in the *gradient space*. This space is spanned by the gradient of the surface height $a(X, Y)$:

$$\boldsymbol{s} = \nabla a = \left[\frac{\partial a}{\partial X}, \frac{\partial a}{\partial Y} \right]^T = \left[s_1, s_2 \right]^T. \tag{8.11}$$

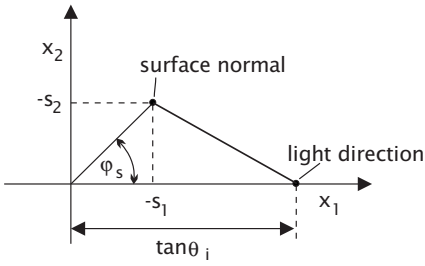

Figure 8.8: *Radiance computation illustrated in the gradient space for a Lambertian surface illuminated by a distant light source with an incidence angle θ_i and an azimuthal angle ϕ_i of zero.*

This gradient is directly related to the surface normal $\boldsymbol{n}$ by

$$\boldsymbol{n} = \left[-\frac{\partial a}{\partial X}, -\frac{\partial a}{\partial Y}, 1 \right]^T = \left[-s_1, -s_2, 1 \right]. \tag{8.12}$$

This equations shows that the gradient space can be understood as a plane parallel to the XY plane at a height $Z = 1$ if we invert the directions of the X and Y axes. The X and Y coordinates where the surface normal vector and other directional vectors intersect this plane are the corresponding coordinates in the gradient space.

The geometry of Lambertian reflection in the gradient space is illustrated in Fig. 8.8. Without loss of generality, we set the direction of the light source as the x direction. Then, the light direction is given by the vector $\boldsymbol{l} = (\tan\theta_i, 0, 1)^T$, and the radiance L of the surface can be expressed as

$$L = \frac{\rho(\lambda)}{\pi} E \frac{\boldsymbol{n}^T \boldsymbol{l}}{|\boldsymbol{n}||\boldsymbol{l}|} = \frac{\rho(\lambda)}{\pi} E \frac{-s_1 \tan\theta_i + 1}{\sqrt{1 + \tan^2\theta_i}\sqrt{1 + s_1^2 + s_2^2}}. \tag{8.13}$$

Contour plots of the radiance distribution in the gradient space are shown in Fig. 8.9a for a light source with an incidence angle of $\theta_i = 0°$. In the case of the light source at the zenith, the contour lines of equal radiance mark lines with constant absolute slope $s = (s_1^2 + s_2^2)^{1/2}$. However, the radiance changes with surface slope are low, especially for low surface inclinations. An oblique illumination leads to a much higher contrast in the radiance (Fig. 8.9b). With an oblique illumination, however, the maximum surface slope in the direction opposite to the light source is limited to $\pi/2 - \theta$ when the surface normal is perpendicular to the light direction.

With a single illumination source, the information about the surface normal is incomplete even if the surface reflectivity is known. Only the component of the surface normal in the direction of the illumination

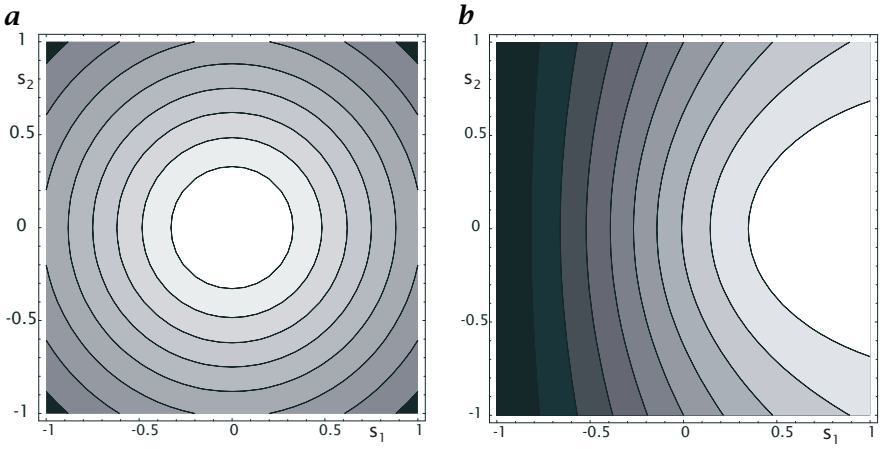

Figure 8.9: *Contour plot of the radiance of a Lambertian surface with homogeneous reflectivity illuminated by parallel light shown in the gradient space for surface slopes between* -1 *and 1. The radiance is normalized to the radiance for a flat surface.* ***a*** *Zero incidence angle* $\theta_i = 0°$*; the spacing of the contour lines is 0.05.* ***b*** *Oblique illumination with an incidence angle of* $45°$ *and an azimuthal angle of* $0°$*; the spacing of the contour lines is 0.1.*

change is given. Thus surface reconstruction with a single illumination source constitutes a complex mathematical problem that will not be considered further here. In the next section we consider how many illuminations from different directions are required to solve the shape from shading problem in a unique way. This technique is known as *photometric stereo*.

8.5.2 Photogrammetric Stereo

The curved contour lines in Fig. 8.9 indicate that the relation between surface slope and radiance is nonlinear. This means that even if we take two different illuminations of the same surface (Fig. 8.10), the surface slope may not be determined in a unique way. This is the case when the curved contour lines intersect each other at more than one point. Only a third exposure with yet another illumination direction would make the solution unique.

Using three exposures also has the significant advantage that the reflectivity of the surface can be eliminated by the use of *ratio imaging*. As an example, we illuminate a Lambertian surface with the same light source from three different directions

$$\begin{aligned} \boldsymbol{l}_1 &= (0,0,1) \\ \boldsymbol{l}_2 &= (\tan\theta_i, 0, 1) \\ \boldsymbol{l}_3 &= (0, \tan\theta_i, 1). \end{aligned} \tag{8.14}$$

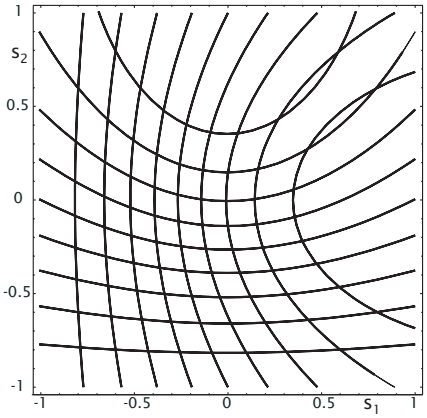

Figure 8.10: *Superimposed contour plots of the radiance of a Lambertian surface with homogeneous reflectivity illuminated by a light source with an angle of incidence of 45° and an azimuthal angle of 0° and 90°, respectively.*

Then

$$L_2/L_1 = \frac{-s_1 \tan\theta_i + 1}{\sqrt{1 + \tan^2\theta_i}}, \quad L_3/L_1 = \frac{-s_2 \tan\theta_i + 1}{\sqrt{1 + \tan^2\theta_i}}. \tag{8.15}$$

Now the equations are linear in s_1 and s_2 and — even better — they are decoupled: s_1 and s_2 depend only on L_2/L_1 and L_3/L_1, respectively (Fig. 8.11). In addition, the normalized radiance in Eq. (8.15) does not depend on the reflectivity of the surface. The reflectivity of the surface is contained in Eq. (8.10) as a factor and thus cancels out when the ratio of two radiance distributions of the same surface is computed.

8.5.3 Shape from Refraction for Specular Surfaces‡

For specular surfaces, the shape from shading techniques discussed in Section 8.5.1 do not work at all as light is only reflected towards the camera when the angle of incidence from the light source is equal to the angle of reflectance. Thus, extended light sources are required. Then, it turns out that for transparent specular surfaces, *shape from refraction* techniques are more advantageous than shape from reflection techniques because the radiance is higher, steeper surface slopes can be measured, and the nonlinearities of the slope/radiance relationship are lower.

A shape from refraction technique requires a special illumination technique, as no significant radiance variations occur, except for the small fraction of light reflected at the surface. The base of the shape from refraction technique is the *telecentric illumination system* which converts a spatial radiance distribution into an angular radiance distribution. Then, all we have to do is to compute the relation between the surface slope and the angle of the refracted beam and to use a light source with an appropriate spatial radiance distribution.

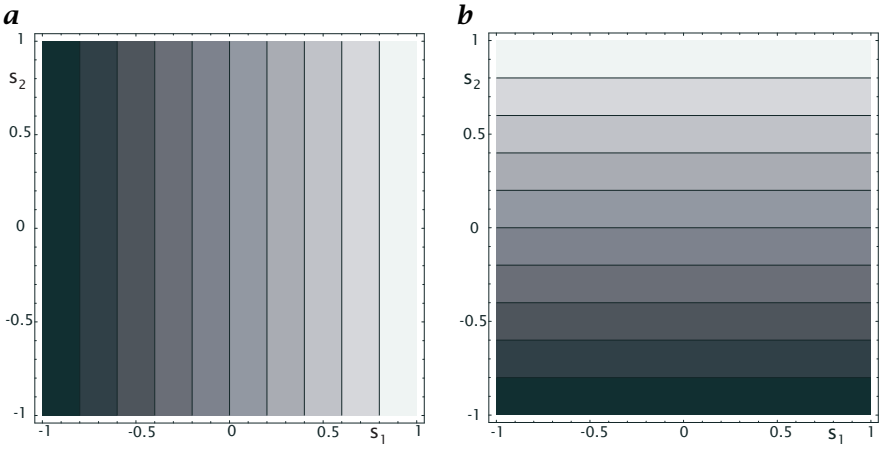

Figure 8.11: *Contour plots of the radiance of a Lambertian surface illuminated by parallel light with an incidence angle of 45° and an azimuthal angle of 0° (**a**) and 90° (**b**), respectively, and normalized by the radiance of the illumination at 0° incidence according to Eq. (8.15). The step size of the contour lines is 0.1. Note the perfect linear relation between the normalized radiance and the x and y surface slope components.*

Figure 8.12 illustrates the optical geometry for the simple case when the camera is placed far above and a light source below a transparent surface of a medium with a higher index of refraction. The relation between the surface slope s and the angle γ is given by Jähne et al. [84] as

$$s = \tan\alpha = \frac{n\tan\gamma}{n - \sqrt{1+\tan^2\gamma}} \approx 4\tan\gamma\left[1 + \frac{3}{2}\tan^2\gamma\right] \tag{8.16}$$

with $n = n_2/n_1$. The inverse relation is

$$\tan\gamma = s\frac{\sqrt{n^2+(n^2-1)s^2}-1}{\sqrt{n^2+(n^2-1)s^2}+s^2} \approx \frac{1}{4}s\left(1 + \frac{3}{32}s^2\right). \tag{8.17}$$

In principle, the shape from refraction technique works for slopes up to infinity (vertical surfaces). In this limiting case, the ray to the camera grazes the surface (Fig. 8.12b) and

$$\tan\gamma = \sqrt{n^2-1}. \tag{8.18}$$

The refraction law thus causes light rays to be inclined in a certain direction relative to the slope of the water surface. If we make the radiance of the light source dependent on the direction of the light beams, the water surface slope becomes visible. The details of the construction of such a system are described by Jähne et al. [84]. Here we just assume that the radiance of the light rays is proportional to $\tan\gamma$ in the x_1 direction. Then we obtain the relation

$$L \propto s_1\frac{\sqrt{n^2+(n^2-1)s^2}-1}{\sqrt{n^2+(n^2-1)s^2}+s^2}. \tag{8.19}$$

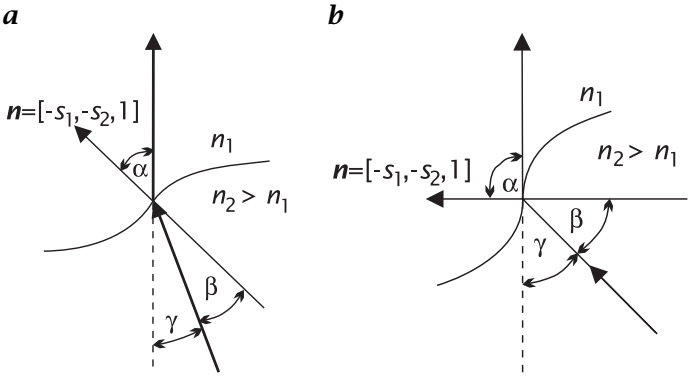

Figure 8.12: *Refraction at an inclined surface as the basis for the shape from refraction technique. The camera is far above the surface.* ***a*** *Rays emitted by the light source at an angle γ are refracted in the direction of the camera.* ***b*** *Even for a slope of infinity (vertical surface, $\alpha = 90°$), rays from the light source meet the camera.*

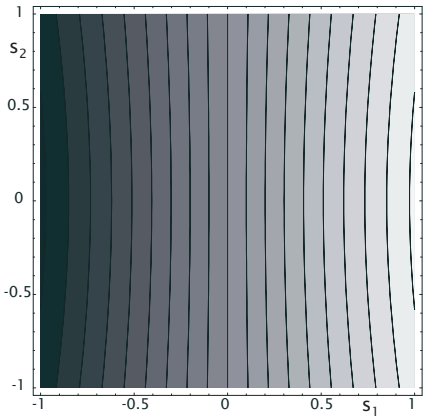

Figure 8.13: *Radiance map for the shape from refraction technique where the radiance in a telecentric illumination source varies linearly in the x_1 direction.*

Of course, again we have the problem that from a scalar quantity such as the radiance no vector component such as the slope can be inferred. The shape from refraction technique, however, comes very close to an ideal setup. If the radiance varies only linearly in the x_1 direction, as assumed, the radiance map in the gradient space is also almost linear (Fig. 8.13). A slight influence of the cross slope (resulting from the nonlinear terms in Eq. (8.19) in s^2) becomes apparent only at quite high slopes.

Ratio imaging can also be used with the shape from refraction technique. Color images have three independent primary colors: red, green, and blue. With a

total of three channels, we can identify the position in a telecentric illumination system — and thus the inclination of the water surface — uniquely and still have one degree of freedom left for corrections. With color imaging we also have the advantage that all three illuminations are taken simultaneously. Thus moving objects can also be observed.

A unique position coding with color can be achieved, for example, with the following color wedges:

$$\begin{aligned} G(\boldsymbol{s}) &= (1/2 + cs_1)E_0(\boldsymbol{s}) \\ R(\boldsymbol{s}) &= [1/2 - c/2(s_1 + s_2)]E_0(\boldsymbol{s}) \\ B(\boldsymbol{s}) &= [1/2 - c/2(s_1 - s_2)]E_0(\boldsymbol{s}). \end{aligned} \tag{8.20}$$

We have again assumed a linear relation between one component of the slope and the radiance, with nonlinear isotropic corrections of the form $s_1 E_0(\boldsymbol{s})$; c is a calibration factor relating the measured radiance to the surface slope.

We now have three illuminations to determine two slope components. Thus, we can take one to compensate for unwanted spatial variation of E_0. This can be done by normalizing the three color channels by the sum of all channels $G + R + B$:

$$\begin{aligned} \frac{G}{G+R+B} &= \frac{2}{3}\left(\frac{1}{2} + cs_1\right), \\ \frac{B-R}{G+R+B} &= \frac{2}{3}cs_2. \end{aligned} \tag{8.21}$$

Then the position on the wedge from which the light originates is given as

$$s_1 = \frac{1}{2c}\frac{2G-R-B}{G+R+B}, \quad s_2 = \frac{3}{2c}\frac{B-R}{G+R+B}. \tag{8.22}$$

From these position values, the x and y components of the slope can be computed according to Eq. (8.19).

8.6 Depth from Multiple Projections: Tomography

8.6.1 Principle

Tomographic methods do not generate a 3-D image of an object directly, but allow reconstruction of the 3-D shape of objects using suitable methods. Tomographic methods can be considered as an extension of stereoscopy. With stereoscopy only the depth of surfaces can be inferred, but not the 3-D shape of transparent objects. Intuitively, we may assume that it is necessary to view such an object from as many directions as possible.

Tomographic methods use radiation that penetrates an object from different directions. If we use a point source (Fig. 8.14b), we observe a perspective or *fan-beam projection* on the screen behind the object just as in optical imaging (Section 7.3). Such an image is taken from different projection directions by rotating the point source and the projection

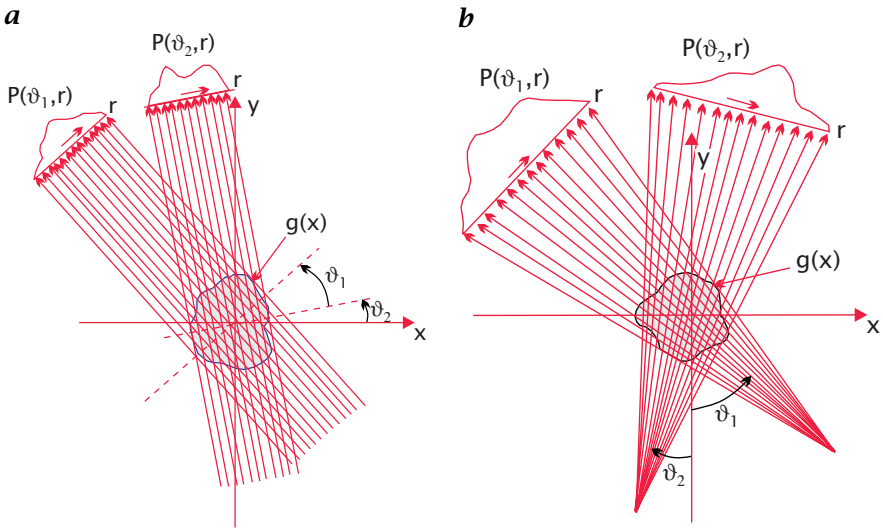

Figure 8.14: ***a*** *Parallel projection and* ***b*** *fan-beam projection in tomography.*

screen around the object. In a similar way, we can use parallel projection (Fig. 8.14a) which is easier to analyze but harder to realize. If the object absorbs the radiation, the intensity loss measured in the projection on the screen is proportional to the path length of the ray in the object. The 3-D shape of the object cannot be reconstructed from one projection. It is necessary to measure projections from all directions by turning the radiation source and projection screen around the object.

As in other imaging methods, tomography can make use of different interactions between matter and radiation. The most widespread application is *transmission tomography*. The imaging mechanism is by the absorption of radiation, e. g., x-rays. Other methods include emission tomography, reflection tomography, and time-of-flight tomography (especially with ultrasound), and complex imaging methods using *magnetic resonance* (*MR*).

8.6.2 Radon Transform and Fourier Slice Theorem

With respect to reconstruction, it is important to note that the projections under all the angles ϑ can be regarded as another 2-D representation of the image. One coordinate is the position in the projection profile, r, the other the angle ϑ (Fig. 8.15). Consequently, we can regard the parallel projection as a transformation of the image into another 2-D representation. Reconstruction then just means applying the inverse transformation. The critical issue, therefore, is to describe the

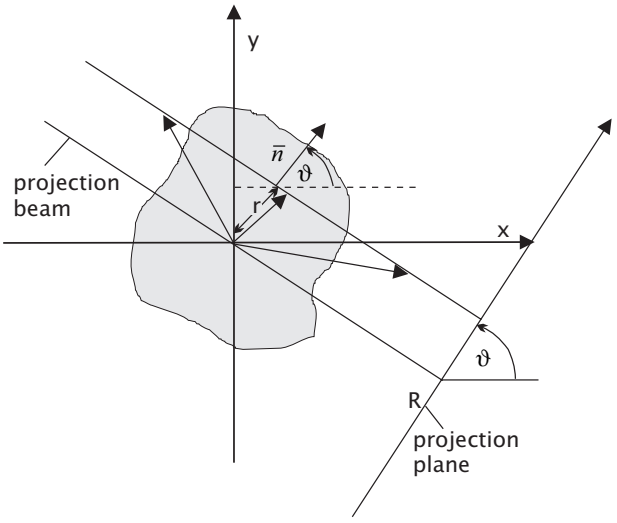

Figure 8.15: *Geometry of a projection beam.*

tomographic transform mathematically and to investigate whether the inverse transform exists.

A projection beam is characterized by the angle ϑ and the offset r (Fig. 8.15). The angle ϑ is the angle between the projection plane and the x axis. Furthermore, we assume that we slice the 3-D object parallel to the xy plane. Then, the scalar product between a vector $\boldsymbol{x}$ on the projection beam and a unit vector

$$\bar{\boldsymbol{n}} = [\cos\vartheta, \sin\vartheta]^T \tag{8.23}$$

normal to the projection beam is constant and equal to the offset r of the beam

$$\boldsymbol{x}\bar{\boldsymbol{n}} - r = x\cos\vartheta + y\sin\vartheta - r = 0. \tag{8.24}$$

The projected intensity $P(r,\vartheta)$ is given by integration along the projection beam:

$$P(r,\vartheta) = \int_{\text{path}} g(\boldsymbol{x})\mathrm{d}\boldsymbol{s} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(\boldsymbol{x})\delta(x_1\cos\vartheta + x_2\sin\vartheta - r)\mathrm{d}^2x. \tag{8.25}$$

The projective transformation of a 2-D function $g(\boldsymbol{x})$ onto $P(r,\vartheta)$ is named after the mathematician Radon as the *Radon transform*.

To better understand the properties of the Radon transform, we analyze it in the Fourier space. The Radon transform can be understood as a special case of a linear shift-invariant filter operation, the *projection operator*. All gray values along the projection beam are added up.

Therefore the point spread function of the projection operator is a δ line in the direction of the projection beam. In the Fourier domain this convolution operation corresponds to a multiplication with the transfer function, which is a δ line (2-D) or δ plane (3-D) normal to the δ line in the spatial domain (see ≻R5). In this way, the projection operator slices a line or plane out of the spectrum that is perpendicular to the projection beam.

This elementary relation can be computed most easily, without loss of generality, in a rotated coordinate system in which the projection direction coincides with the y' axis. Then the r coordinate in $P(r,\vartheta)$ coincides with the x' coordinate and ϑ becomes zero. In this special case, the Radon transform reduces to an integration along the y' direction:

$$P(x',0) = \int_{-\infty}^{\infty} g(x',y')\,\mathrm{d}y'. \tag{8.26}$$

The Fourier transform of the projection function can be written as

$$\hat{P}(k_{x'},0) = \int_{-\infty}^{\infty} P(x',0)\exp(-2\pi\mathrm{i}k_{x'}x')\,\mathrm{d}x'. \tag{8.27}$$

Replacing $P(x',0)$ by the definition of the Radon transform, Eq. (8.26) yields

$$\hat{P}(k_{x'},0) = \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} g(x',y')\,\mathrm{d}y'\right]\exp(-2\pi\mathrm{i}k_{x'}x')\,\mathrm{d}x'. \tag{8.28}$$

If we insert the factor $\exp(-2\pi\mathrm{i}0y') = 1$ in this double integral, we recognize that the integral is a 2-D Fourier transform of $g(x',y')$ for $k_{y'} = 0$:

$$\begin{aligned}\hat{P}(k_{x'},0) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x',y')\exp(-2\pi\mathrm{i}k_{x'}x')\exp(-2\pi\mathrm{i}0y')\,\mathrm{d}x'\,\mathrm{d}y' \\ &= g(k_{x'},0).\end{aligned} \tag{8.29}$$

Back transformation into the original coordinate system finally yields

$$\hat{P}(q,\vartheta) = \hat{g}(\boldsymbol{k})\delta(\boldsymbol{k} - (\boldsymbol{k}\bar{\boldsymbol{n}})\bar{\boldsymbol{n}}), \tag{8.30}$$

where q is the coordinate in the k space in the direction of ϑ and $\bar{\boldsymbol{n}}$ the normal vector introduced in Eq. (8.23). The spectrum of the projection is identical to the spectrum of the original object on a beam normal to the direction of the projection beam. This important result is called the *Fourier slice theorem* or *projection theorem*.

8.6.3 Filtered Back-Projection

If the projections from all directions are available, the slices of the spectrum obtained cover the complete spectrum of the object. Inverse Fourier transform then yields the original object. Filtered back-projection uses this approach with a slight modification. If we just added the spectra of the individual projection beams to obtain the complete spectrum of the object, the spectral density for small wave numbers would be too high as the beams are closer to each other for small radii. Thus, we must correct the spectrum with a suitable weighting factor. In the continuous case, the geometry is very easy. The density of the projection beams goes with $|\boldsymbol{k}|^{-1}$. Consequently, the spectra of the projection beams must be multiplied by $|\boldsymbol{k}|$. Thus, filtered back-projection is a two-step process. First, the individual projections must be filled before the reconstruction can be performed by summing up the back-projections.

In the first step, we thus multiply the spectrum of each projection direction by a suitable weighting function $\hat{w}(|\boldsymbol{k}|)$. Of course, this operation can also be performed as a convolution with the inverse Fourier transform of $\hat{w}(|\boldsymbol{k}|)$, $w(r)$. Because of this step, the procedure is called the *filtered back-projection*.

In the second step, the back-projection is performed and each projection gives a slice of the spectrum. Adding up all the filtered spectra yields the complete spectrum. As the Fourier transform is a linear operation, we can add up the filtered projections in the space domain. In the space domain, each filtered projection contains the part of the object that is constant in the direction of the projection beam. Thus, we can back-project the corresponding gray value of the filtered projection along the direction of the projection beam and add it up to the contributions from the other projection beams.

After this illustrative description of the principle of the filtered back-projection algorithm we derive the method for the continuous case. We start with the Fourier transform of the object and write the inverse Fourier transformation in polar coordinates (q, ϑ) in order to make use of the Fourier slice theorem

$$g(\boldsymbol{x}) = \int_0^{2\pi} \int_0^{\infty} q\hat{g}(q,\vartheta) \exp[\mathrm{i}q(x_1 \cos\vartheta + x_2 \sin\vartheta)] \mathrm{d}q \mathrm{d}\theta. \tag{8.31}$$

In this formula, the spectrum is already multiplied by the wave number, q. The integration boundaries, however, are not yet correct to be applied to the Fourier slice theorem (Eq. (8.30)). The coordinate, q, should run from $-\infty$ to ∞ and ϑ only from 0 to π. In Eq. (8.31), we integrate only over half a beam from the origin to infinity. We can compose a full beam from two half beams at the angles ϑ and $\vartheta + \pi$. Thus, we split the integral

in Eq. (8.31) into two over the angle ranges $[0, \pi[$ and $[\pi, 2\pi[$ and obtain

$$\begin{aligned} g(\boldsymbol{x}) &= \int_0^{\pi}\int_0^{\infty} q\hat{g}(q,\vartheta)\exp[\mathrm{i}q(x_1\cos\vartheta + x_2\sin\vartheta)]\,\mathrm{d}q\,\mathrm{d}\vartheta \\ &+ \int_0^{\pi}\int_0^{\infty} q\hat{g}(-q,\vartheta')\exp[-\mathrm{i}q(x_1\cos\vartheta' + x_2\sin\vartheta')]\,\mathrm{d}q\,\mathrm{d}\vartheta' \end{aligned}$$

using the following identities:

$$\vartheta' = \vartheta + \pi,\ \hat{g}(-r,\vartheta) = \hat{g}(r,\vartheta'),\ \cos(\vartheta') = -\cos(\vartheta),\ \sin(\vartheta') = -\sin(\vartheta).$$

Now we can recompose the two integrals again, if we substitute q by $-q$ in the second integral and replace $\hat{g}(q,\vartheta)$ by $\hat{P}(q,\vartheta)$ because of the Fourier slice theorem Eq. (8.30):

$$g(\boldsymbol{x}) = \int_0^{\pi}\int_{-\infty}^{\infty} |q|\hat{P}(q,\vartheta)\exp[\mathrm{i}q(x_1\cos\vartheta + x_2\sin\vartheta)]\,\mathrm{d}q\,\mathrm{d}\vartheta. \tag{8.32}$$

Equation (8.32) gives the inverse Radon transform and is the basis for the *filtered back-projection* algorithm. The inner integral performs the back-projection of a single projection:

$$P' = \mathcal{F}^{-1}(|q|\mathcal{F}P). \tag{8.33}$$

$\mathcal{F}$ denotes the 1-D Fourier transform operator. P' is the projection function P multiplied in the Fourier space by $|\boldsymbol{q}|$. If we perform this operation as a convolution in the space domain, we can formally write

$$P' = [\mathcal{F}^{-1}(|q|)] * P. \tag{8.34}$$

The outer integral in Eq. (8.32) over the angle ϑ,

$$g(\boldsymbol{x}) = \int_0^{\pi} P'(r,\vartheta)\,\mathrm{d}\vartheta, \tag{8.35}$$

sums up the back-projected and filtered projections over all directions and thus forms the reconstructed image. Note that the filtered projection profile $P'(r,\vartheta)$ in Eq. (8.35) must be regarded as a 2-D function to built up a 2-D object $g(\boldsymbol{x})$. This means that the projection profile is projected back into the projection direction.

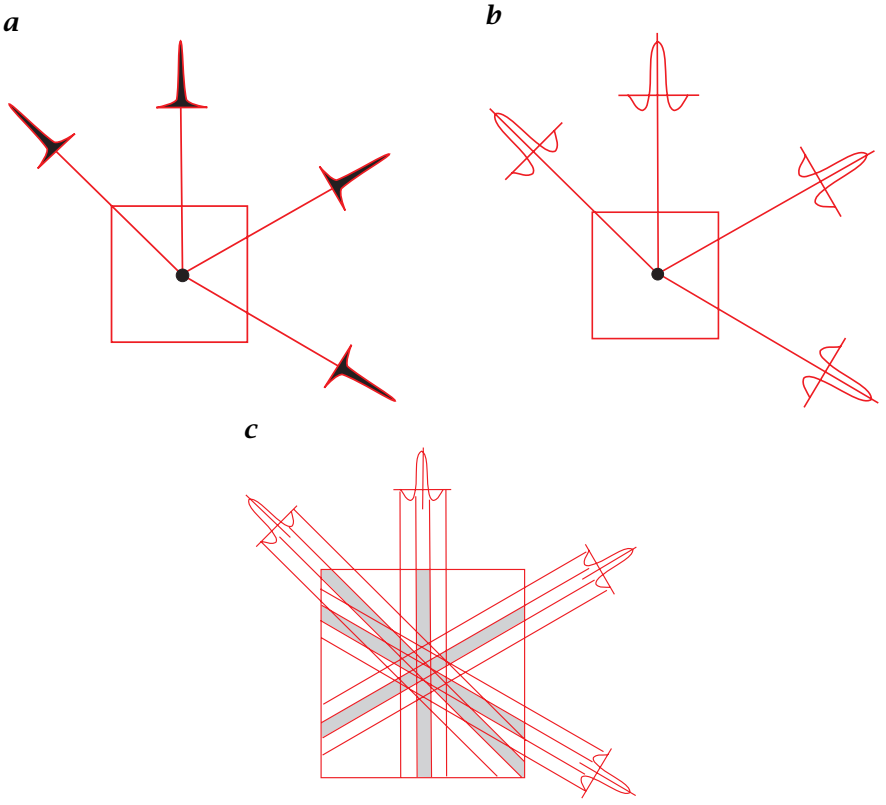

Figure 8.16: *Illustration of the filtered back-projection algorithm with a point object:* ***a*** *projections from different directions;* ***b*** *filtering of the projection functions;* ***c*** *back-projection: adding up the filtered projections.*

8.6.4 Discrete Filtered Back-Projection

There are several details we have not yet discussed that cause serious problems for the reconstruction in the infinite continuous case. First, we observe that it is impossible to reconstruct the mean of an object. Because of the multiplication by $|\boldsymbol{k}|$ in the Fourier domain (Eq. (8.32)), $\hat{g}(\mathbf{0})$ is eliminated. Second, it is altogether impossible to reconstruct an object of infinite size, as any projection beam will result in infinite values.

Fortunately, all these difficulties disappear where we turn from the infinite continuous case to the finite discrete case where the objects are of limited size. In practice, the size limit is given by the distance between the radiation source and the detector. The resolution of the projection profile is limited by the combined effects of the extent of the radiation source and the resolution of the detector array in the projection plane.

Finally, we can only take a limited number of projections. This corresponds to a sampling of the angle ϑ in the Radon representation of the image.

We illustrate the discussion in this section with an example. We can learn much about projection and reconstruction by considering the reconstruction of the simplest object, a point, because the Radon transform (Eq. (8.25)) and its inverse are linear transforms. Then, the projections from all directions are equal (Fig. 8.16a) and show a sharp maximum in the projection functions $P(r, \vartheta_i)$. In the first step of the filtered back-projection algorithm, P is convolved with the $|k|$ filter. The result is a modified projection function P' which is identical to the point spread function of the $|k|$ filter (Fig. 8.16b).

In a second step, the back-projections are added up in the image. From Fig. 8.16c, we can see that at the position of the point in the image the peaks from all projections add up. At all other positions in the images, the filtered back-projections are superimposed on each other in a destructive manner, because they show negative and positive values. If the projection directions are sufficiently close to each other, they cancel each other out except for the point at the center of the image. Figure 8.16c also demonstrates that an insufficient number of projections leads to star-shaped distortion patterns.

The simple example of the reconstruction of a point from its projections is also useful to show the importance of filtering the projections. Let us imagine what happens when we omit this step. Then, we would add up δ lines as back-projections which rotate around the position of the point. Consequently, we would not obtain a point but a rotation-symmetric function that falls off with $|\boldsymbol{x}|^{-1}$. As a result, the reconstructed objects would be considerably blurred.

8.7 Further Readings‡

A whole part with seven chapters of the "Handbook of Computer Vision and Applications" is devoted to 3-D imaging [83, Vol. I, Part IV]. Klette et al. [97] discusses 3-D computer vision focusing on stereo, shape from shading, and photometric stereo.

9 Digitization, Sampling, Quantization

9.1 Definition and Effects of Digitization

The final step of digital image formation is the *digitization*. This means sampling the gray values at a discrete set of points, which can be represented by a matrix. Sampling may already occur in the sensor that converts the collected photons into an electrical signal. In a conventional tube camera, the image is already sampled in lines, as an electron beam scans the imaging tube line by line. A CCD camera already has a matrix of discrete sensors. Each sensor is a sampling point on a 2-D grid. The standard video signal, however, is again an analog signal. Consequently, we lose the horizontal sampling, as the signal from a line of sensors is converted back to an analog signal.

At first glance, digitization of a continuous image appears to be an enormous loss of information, because a continuous function is reduced to a function on a grid of points. Therefore the crucial question arises as to which criterion we can use to ensure that the sampled points are a valid representation of the continuous image, i. e., there is no loss of information. We also want to know how and to which extent we can reconstruct a continuous image from the sampled points. We will approach these questions by first illustrating the distortions that result from improper sampling.

Intuitively, it is clear that sampling leads to a reduction in resolution, i. e., structures of about the scale of the sampling distance and finer will be lost. It might come as a surprise to know that considerable distortions occur if we sample an image that contains fine structures. Figure 9.1 shows a simple example. Digitization is simulated by overlaying a 2-D grid on the object comprising two linear grids with different grid constants. After sampling, both grids appear to have grid constants with different periodicity and direction. This kind of image distortion is called the *Moiré effect*.

The same phenomenon, called *aliasing*, is known for one-dimensional signals, especially time series. Figure 9.2 shows a signal with a sinusoidal oscillation. It is sampled with a sampling distance which is slightly smaller than its wavelength. As a result we will observe a much larger wavelength. Whenever we digitize analog data, these problems occur. It is a general phenomenon of signal processing. In this respect, im-

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2

Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

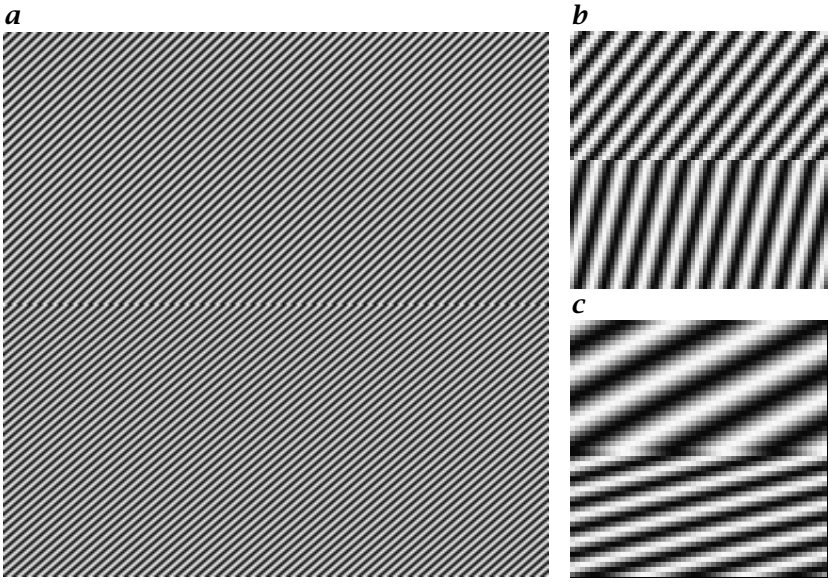

Figure 9.1: *The Moiré effect.* ***a*** *Original image with two periodic patterns: top* $\mathbf{k} = [0.21, 0.22]^T$*, bottom* $\mathbf{k} = [0.21, 0.24]^T$*.* ***b*** *Each fourth and* ***c*** *each fifth point are sampled in each direction, respectively.*

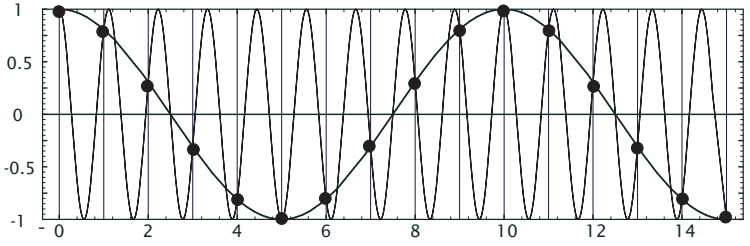

Figure 9.2: *Demonstration of the aliasing effect: an oscillatory signal is sampled with a sampling distance* Δx *equal to 9/10 of the wavelength. The result is an aliased wavelength which is 10 times the sampling distance.*

age processing is only a special case in the more general field of signal theory.

Because the aliasing effect has been demonstrated with periodic signals, the key to understand and thus to avoid it is to analyze the digitization process in Fourier space. In the following, we will perform this analysis step by step. As a result, we can formulate the conditions under which the sampled points are a correct and complete representation of the continuous image in the so-called *sampling theorem*. The following considerations are not a strict mathematical proof of the sampling theorem but rather an illustrative approach.

9.2 Image Formation, Sampling, Windowing

Our starting point is an infinite, continuous image $g(\boldsymbol{x})$, which we want to map onto a matrix $\boldsymbol{G}$. In this procedure we will include the image formation process, which we discussed in Section 7.6. We can then distinguish three separate steps: image formation, sampling, and the limitation to a finite image matrix.

9.2.1 Image Formation

Digitization cannot be treated without the image formation process. The optical system, including the sensor, influences the image signal so that we should include this process.

Digitization means that we sample the image at certain points of a discrete grid, $\boldsymbol{r}_{m,n}$ (Section 2.2.3). If we restrict our considerations to rectangular grids, these points can be written according to Eq. (2.2):

$$\boldsymbol{r}_{m,n} = [m\,\Delta x_1, n\,\Delta x_2]^T \quad \text{with} \quad m,n \in \mathbb{Z}. \tag{9.1}$$

Generally, we do not collect the illumination intensity exactly at these points, but in a certain area around them. As an example, we take an ideal CCD camera, which consists of a matrix of photodiodes without any light-insensitive strips in between. We further assume that the photodiodes are equally sensitive over the whole area. Then the signal at the grid points is the integral over the area of the individual photodiodes:

$$g(\boldsymbol{r}_{m,n}) = \int_{(m-1/2)\Delta x_1}^{(m+1/2)\Delta x_1} \int_{(n-1/2)\Delta x_2}^{(n+1/2)\Delta x_2} g'(\boldsymbol{x})\,\mathrm{d}x_1\,\mathrm{d}x_2. \tag{9.2}$$

This operation includes *convolution* with a rectangular box function and sampling at the points of the grid. These two steps can be separated. We can perform first the continuous convolution and then the sampling. In this way we can generalize the image formation process and separate it from the sampling process.

Because convolution is an associative operation, we can combine the averaging process of the CCD sensor with the PSF of the optical system (Section 7.6.1) in a single convolution process. Therefore, we can describe the image formation process in the spatial and Fourier domain by the following operation:

$$g(\boldsymbol{x}) = \int_{-\infty}^{\infty} g'(\boldsymbol{x}')h(\boldsymbol{x}-\boldsymbol{x}')\mathrm{d}^2x' \quad \circ\!\!-\!\!\bullet \quad \hat{g}(\boldsymbol{k}) = \hat{g}'(\boldsymbol{k})\hat{h}(\boldsymbol{k}), \tag{9.3}$$

where $h(\boldsymbol{x})$ and $\hat{h}(\boldsymbol{k})$ are the resulting PSF and OTF, respectively, and $g'(\boldsymbol{x})$ can be considered as the gray value image that would be obtained

by a perfect sensor, i. e., an optical system (including the sensor) whose OTF is identically 1 and whose PSF is a δ-function.

Generally, the *image formation* process results in a blurring of the image; fine details are lost. In Fourier space this leads to an attenuation of high wave numbers. The resulting gray value image is said to be *band-limited*.

9.2.2 Sampling

Now we perform the *sampling*. Sampling means that all information is lost except at the grid points. Mathematically, this constitutes a multiplication of the continuous function with a function that is zero everywhere except for the grid points. This operation can be performed by multiplying the image function $g(\boldsymbol{x})$ with the sum of δ functions located at the grid points $\boldsymbol{r}_{m,n}$ Eq. (9.1). This function is called the two-dimensional δ comb, or "*bed-of-nails function*". Then sampling can be expressed as

$$g_s(\boldsymbol{x}) = g(\boldsymbol{x}) \sum_{m,n} \delta(\boldsymbol{x} - \boldsymbol{r}_{m,n}) \quad \circ\!\!-\!\!\bullet \quad \hat{g}_s(\boldsymbol{k}) = \sum_{p,q} \hat{g}(\boldsymbol{k} - \hat{\boldsymbol{r}}_{p,q}), \tag{9.4}$$

wobei

$$\hat{\boldsymbol{r}}_{p,q} = \begin{bmatrix} p \square k_1 \\ q \square k_2 \end{bmatrix} \quad \text{with} \quad p, q \in \mathbb{Z} \quad \text{and} \quad \square k_w = \frac{1}{\Delta x_w} \tag{9.5}$$

are the points of the so-called *reciprocal grid*, which plays a significant role in solid state physics and crystallography. According to the convolution theorem (Theorem 4, p. 52), multiplication of the image with the 2-D δ comb corresponds to a convolution of the Fourier transform of the image, the image spectrum, with another 2-D δ comb, whose grid constants are reciprocal to the grid constants in x space (see Eqs. (9.1) and (9.5)). A dense sampling in x space yields a wide mesh in the k space, and vice versa. Consequently, sampling results in a reproduction of the image spectrum at each grid point $\hat{\boldsymbol{r}}_{p,q}$ in the Fourier space.

9.2.3 Sampling Theorem

Now we can formulate the condition where we get no distortion of the signal by sampling, known as the *sampling theorem*. If the image spectrum is so extended that parts of it overlap with the periodically repeated copies, then the overlapping parts are alternated. We cannot distinguish whether the spectral amplitudes come from the original spectrum at the center or from one of the copies. In order to obtain no distortions, we must avoid overlapping.

A safe condition to avoid overlapping is as follows: the spectrum must be restricted to the area that extends around the central grid point

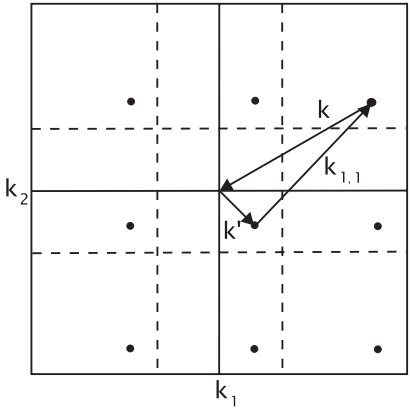

Figure 9.3: *Explanation of the Moiré effect with a periodic structure that does not meet the sampling condition.*

up to the lines parting the area between the central grid point and all other grid points. In solid state physics this zone is called the first Brillouin zone [96].

On a rectangular W-dimensional grid, this results in the simple condition that the maximum wave number at which the image spectrum is not equal to zero must be restricted to less than half of the grid constants of the reciprocal grid:

Theorem 10 (Sampling theorem) *If the spectrum $\hat{g}(\boldsymbol{k})$ of a continuous function $g(\boldsymbol{x})$ is band-limited, i.e.,*

$$\hat{g}(\boldsymbol{k}) = 0 \quad \forall \; |k_w| \geq \square k_w/2, \tag{9.6}$$

then it can be reconstructed exactly from samples with a distance

$$\Delta x_w = 1/\square k_w. \tag{9.7}$$

In other words, we will obtain a periodic structure correctly only if we take at least two samples per wavelength. The maximum wave number that can be sampled without errors is called the *Nyquist* or *limiting* wave number. In the following, we will often use dimensionless wave numbers which are scaled to the limiting wave number. We denote this scaling with a tilde:

$$\tilde{k}_w = \frac{k_w}{\square k_w/2} = 2k_w \Delta x_w. \tag{9.8}$$

In this scaling all the components of the wave number $\tilde{k}_w$ fall into the $]-1, 1[$ interval.

Now we can explain the Moiré and aliasing effects. We start with a periodic structure that does not meet the sampling condition. The

original spectrum contains a single peak, which is marked with the long vector $\boldsymbol{k}$ in Fig. 9.3.

Because of the periodic replication of the sampled spectrum, there is exactly one peak, at $\boldsymbol{k}'$, which lies in the central cell. Figure 9.3 shows that this peak has not only another wavelength but in general another direction, as observed in Fig. 9.1.

The observed wave number $\boldsymbol{k}'$ differs from the true wave number $\boldsymbol{k}$ by a grid translation vector $\hat{\boldsymbol{r}}_{p,q}$ on the reciprocal grid. The indices p and q must be chosen to meet the condition

$$\begin{aligned} |k_1 + p\ \square k_1| &< \square k_1/2 \\ |k_2 + q\ \square k_2| &< \square k_2/2. \end{aligned} \tag{9.9}$$

According to this condition, we obtain an aliased wave number

$$k_1' = k_1 - \square k_1 = 9/10\ \square k_1 - \square k_1 = -1/10\ \square k_1 \tag{9.10}$$

for the one-dimensional example in Fig. 9.2, as we just observed.

The sampling theorem, as formulated above, is actually too strict a requirement. A sufficient and necessary condition is that the periodic replications of the image spectra must not overlap.

9.2.4 Limitation to a Finite Window

So far, the sampled image is still infinite in size. In practice, we can only work with finite image matrices. Thus the last step is the limitation of the image to a finite window size. The simplest case is the multiplication of the sampled image with a box function. More generally, we can take any *window function* $g_l(\boldsymbol{x})$ which is zero for sufficiently large $\boldsymbol{x}$ values:

$$g_l(\boldsymbol{x}) = g_s(\boldsymbol{x}) \cdot w(\boldsymbol{x}) \quad \circ\!\!-\!\!\bullet \quad \hat{g}_l(\boldsymbol{k}) = \hat{g}_s(\boldsymbol{k}) * \hat{w}(\boldsymbol{k}). \tag{9.11}$$

In Fourier space, the spectrum of the sampled image will be convolved with the Fourier transform of the window function. Let us consider the example of the box window function in detail. If the window in the x space includes $M \times N$ sampling points, its size is $M\Delta x_1 \times N\Delta x_2$. The Fourier transform of the 2-D box function is the 2-D sinc function (≻ R5). The main peak of the sinc function has a half-width of $2\pi/(M\Delta x_1) \times 2\pi/(N\Delta x_2)$. A narrow peak in the spectrum of the image will become a 2-D sinc function. Generally, the resolution in the spectrum will be reduced to the order of the half-width of the sinc function.

In summary, sampling leads to a limitation of the wave number, while the limitation of the image size determines the wave number resolution. Thus the scales in space and wave number domains are reciprocal to each other. The resolution in the space domain determines the size in the wave number domain, and vice versa.

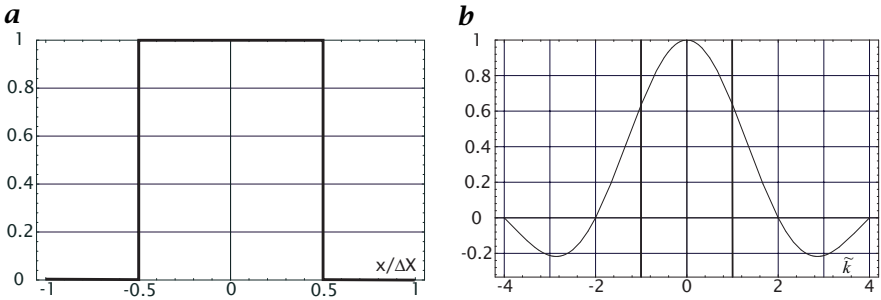

Figure 9.4: ***a*** *PSF and* ***b*** *transfer function of standard sampling.*

9.2.5 Standard Sampling

The type of sampling discussed in Section 9.2.1 using the example of the ideal CCD camera is called *standard sampling*. Here the mean value of an elementary cell is assigned to a corresponding sampling point. It is a kind of *regular* sampling, since each point in the continuous space is equally weighted. We might be tempted to assume that standard sampling conforms to the sampling theorem. Unfortunately, this is not the case (Fig. 9.4). To the Nyquist wave number, the Fourier transform of the box function is still $1/\sqrt{2}$. The first zero crossing occurs at double the Nyquist wave number. Consequently, Moiré effects will be observed with CCD cameras. The effects are even more pronounced as only a small fraction — typically 20% of the chip area for interline transfer cameras — are light sensitive [108].

Smoothing over larger areas with a box window is not of much help as the Fourier transform of the box window only decreases with k^{-1} (Fig. 9.4). The ideal window function for sampling is identical to the ideal interpolation formula Eq. (9.15) discussed in Section 9.3, as its Fourier transform is a box function with the width of the elementary cell of the reciprocal grid. However, this windowing is impracticable. A detailed discussion of interpolation can be found in Section 10.6.

9.3 Reconstruction from Samples†

9.3.1 Perfect Reconstruction

The sampling theorem ensures the conditions under which we can reconstruct a continuous function from sampled points, but we still do not know how to perform the reconstruction of the continuous image from its samples, i. e., the inverse operation to sampling.

Reconstruction is performed by a suitable *interpolation* of the sampled points. Generally, the interpolated points $g_r(\boldsymbol{x})$ are calculated from

the sampled values $g(\boldsymbol{r}_{m,n})$ weighted with suitable factors depending on the distance from the interpolated point:

$$g_r(\boldsymbol{x}) = \sum_{m,n} h(\boldsymbol{x} - \boldsymbol{r}_{m,n}) g_s(\boldsymbol{r}_{m,n}). \tag{9.12}$$

Using the integral properties of the δ function, we can substitute the sampled points on the right side by the continuous values:

$$\begin{aligned} g_r(\boldsymbol{x}) &= \sum_{m,n} \int_{-\infty}^{\infty} h(\boldsymbol{x} - \boldsymbol{x}') g(\boldsymbol{x}') \delta(\boldsymbol{r}_{m,n} - \boldsymbol{x}') \mathrm{d}^2 x' \\ &= \int_{-\infty}^{\infty} h(\boldsymbol{x} - \boldsymbol{x}') \sum_{m,n} \delta(\boldsymbol{r}_{m,n} - \boldsymbol{x}') g(\boldsymbol{x}') \mathrm{d}^2 x'. \end{aligned}$$

The latter integral is a convolution of the weighting function h with the product of the image function g and the 2-D δ-comb. In Fourier space, convolution is replaced by complex multiplication and vice versa:

$$\hat{g}_r(\boldsymbol{k}) = \hat{h}(\boldsymbol{k}) \sum_{u,v} \hat{g}(\boldsymbol{k} - \hat{\boldsymbol{r}}_{u,v}). \tag{9.13}$$

The interpolated function cannot be equal to the original image if the periodically repeated image spectra are overlapping. This is nothing new; it is exactly what the sampling theorem states. The interpolated image function is only equal to the original image function if the weighting function is a box function with the width of the elementary cell of the reciprocal grid. Then the effects of the sampling — all replicated and shifted spectra — are eliminated and only the original band-limited spectrum remains, and Eq. (9.13) becomes:

$$\hat{g}_r(\boldsymbol{k}) = \Pi(k_1 \Delta x_1 / 2\pi, k_2 \Delta x_2 / 2\pi) \hat{g}(\boldsymbol{k}). \tag{9.14}$$

Then the interpolation function is the inverse Fourier transform of the box function

$$h(\boldsymbol{x}) = \frac{\sin \pi x_1 / \Delta x_1}{\pi x_1 / \Delta x_1} \frac{\sin \pi x_2 / \Delta x_2}{\pi x_2 / \Delta x_2}. \tag{9.15}$$

Unfortunately, this function decreases only with $1/x$ towards zero. Therefore, a correct interpolation requires a large image area; mathematically, it must be infinitely large. This condition can be weakened if we "overfill" the sampling theorem, i. e., ensure that $\hat{g}(\boldsymbol{k})$ is already zero before we reach the Nyquist wave number. According to Eq. (9.13), we can then choose $\hat{h}(\boldsymbol{k})$ arbitrarily in the region where $\hat{g}$ vanishes. We can use this freedom to construct an interpolation function that decreases more quickly in the spatial domain, i. e., has a minimum-length interpolation mask. We can also start from a given interpolation formula.

Then the deviation of its Fourier transform from a box function tells us to what extent structures will be distorted as a function of the wave number. Suitable interpolation functions will be discussed in detail in Section 10.6.

9.3.2 Multidimensional Sampling on Nonorthogonal Grids‡

So far, sampling has only been considered for rectangular 2-D grids. Here we will see that it can easily be extended to higher dimensions and nonorthogonal grids. Two extensions are required.

First, W-dimensional grid vectors must be defined using a set of W not necessarily orthogonal basis vectors $\boldsymbol{b}_w$ that span the W-dimensional space. Then a vector on the lattice is given by

$$\boldsymbol{r}_{\boldsymbol{n}} = [n_1\boldsymbol{b}_1, n_2\boldsymbol{b}_2, \ldots, n_W\boldsymbol{b}_W]^T \quad \text{with} \quad \boldsymbol{n} = [n_1, n_2, \ldots, n_W],\ n_w \in \mathbb{Z}. \tag{9.16}$$

In image sequences one of these coordinates is the time. Second, for some types of lattices, e. g., a triangular grid, more than one point is required. Thus for general regular lattices, P points per elementary cell must be considered. Each of the points of the elementary cell is identified by an offset vector $\boldsymbol{s}_p$.

Therefore an additional sum over all points in the elementary cell is required in the sampling integral, and Eq. (9.4) extends to

$$g_s(\boldsymbol{x}) = g(\boldsymbol{x}) \sum_p \sum_{\boldsymbol{n}} \delta(\boldsymbol{x} - \boldsymbol{r}_{\boldsymbol{n}} - \boldsymbol{s}_p). \tag{9.17}$$

In this equation, the summation ranges have been omitted.

The extended sampling theorem directly results from the Fourier transform of Eq. (9.17). In this equation the continuous signal $g(\boldsymbol{x})$ is multiplied by the sum of delta combs. According to the convolution theorem (Theorem 4, p. 52), this results in a convolution of the Fourier transform of the signal and the sum of the delta combs in Fourier space. The Fourier transform of a delta comb is again a delta comb (> R5). As the convolution of a signal with a delta distribution simply replicates the function value at the zero point of the delta functions, the Fourier transform of the sampled signal is simply a sum of shifted copies of the Fourier transform of the signal:

$$\hat{g}_s(\boldsymbol{k}, \nu) = \sum_p \sum_{\boldsymbol{v}} \hat{g}(\boldsymbol{k} - \hat{\boldsymbol{r}}_{\boldsymbol{v}}) \exp(-2\pi \mathrm{i} \boldsymbol{k}^T \boldsymbol{s}_p). \tag{9.18}$$

The phase factor $\exp(-2\pi \mathrm{i} \boldsymbol{k}^T \boldsymbol{s}_p)$ results from the shift of the points in the elementary cell by $\boldsymbol{s}_p$ according to the shift theorem (Theorem 3, p. 52). The vectors $\hat{\boldsymbol{r}}_{\boldsymbol{v}}$

$$\hat{\boldsymbol{r}}_{\boldsymbol{v}} = v_1\hat{\boldsymbol{b}}_1 + v_2\hat{\boldsymbol{b}}_2 + \ldots + v_D\hat{\boldsymbol{b}}_D \quad \text{with} \quad v_d \in \mathbb{Z} \tag{9.19}$$

are the points of the *reciprocal lattice*. The fundamental translation vectors in the space and Fourier domain are related to each other by

$$\boldsymbol{b}_d^T \hat{\boldsymbol{b}}_{d'} = \delta_{d-d'}. \tag{9.20}$$

This basically means that a fundamental translation vector in the Fourier domain is perpendicular to all translation vectors in the spatial domain except

for the corresponding one. Furthermore, the magnitudes of the corresponding vectors are reciprocally related to each other, as their scalar product is one. In 3-D space, the fundamental translations of the reciprocial lattice can therefore be computed by

$$\hat{\boldsymbol{b}}_d = \frac{\boldsymbol{b}_{d+1} \times \boldsymbol{b}_{d+2}}{\boldsymbol{b}_1^T (\boldsymbol{b}_2 \times \boldsymbol{b}_3)}. \tag{9.21}$$

The indices in the preceding equation are computed modulo 3, and $\boldsymbol{b}_1^T(\boldsymbol{b}_2 \times \boldsymbol{b}_3)$ is the volume of the primitive elementary cell in the spatial domain. All these equations are familiar to solid state physicists or crystallographers [96]. Mathematicians know the lattice in the Fourier domain as the *dual base* or *reciprocal base* of a vector space spanned by a nonorthogonal base. For an orthogonal base, all vectors of the dual base show into the same direction as the corresponding vectors and the magnitude is given by $\left|\hat{\boldsymbol{b}}_d\right| = 1/|\boldsymbol{b}_d|$. Then often the length of the base vectors is denoted by Δx_d, and the length of the reciprocal vectors by $\square k_d = 1/\Delta x_d$. Thus an orthonormal base is dual to itself.

Reconstruction of the continuous signal is performed again by a suitable *interpolation* of the values at the sampled points. Now the interpolated values $g_r(\boldsymbol{x})$ are calculated from the values sampled at $\boldsymbol{r_n} + \boldsymbol{s}_p$, weighted with suitable factors that depend on the distance from the interpolated point:

$$g_r(\boldsymbol{x}) = \sum_p \sum_{\boldsymbol{n}} g_s(\boldsymbol{r_n} + \boldsymbol{s}_p) h(\boldsymbol{x} - \boldsymbol{r_n} - \boldsymbol{s}_p). \tag{9.22}$$

Using the integral property of the δ distributions, we can substitute the sampled points on the right-hand side by the continuous values and then interchange summation and integration:

$$\begin{aligned} g_r(\boldsymbol{x}) &= \sum_p \sum_{\boldsymbol{n}} \int_{-\infty}^{\infty} g(\boldsymbol{x}') h(\boldsymbol{x} - \boldsymbol{x}') \delta(\boldsymbol{r_n} + \boldsymbol{s}_p - \boldsymbol{x}') \mathrm{d}^W x' \\ &= \int_{-\infty}^{\infty} h(\boldsymbol{x} - \boldsymbol{x}') \sum_p \sum_{\boldsymbol{n}} \delta(\boldsymbol{r_n} + \boldsymbol{s}_p - \boldsymbol{x}') g(\boldsymbol{x}') \mathrm{d}^W x'. \end{aligned}$$

The latter integral is a convolution of the weighting function h with a function that is the sum of the product of the image function g with shifted δ combs. In Fourier space, convolution is replaced by complex multiplication and vice versa. If we further consider the shift theorem and that the Fourier transform of a δ comb is again a δ comb, we finally obtain

$$\hat{g}_r(\boldsymbol{k}) = \hat{h}(\boldsymbol{k}) \sum_p \sum_{\boldsymbol{v}} \hat{g}(\boldsymbol{k} - \hat{\boldsymbol{r}}_{\boldsymbol{v}}) \exp(-2\pi \mathrm{i} \boldsymbol{k}^T \boldsymbol{s}_p). \tag{9.23}$$

The interpolated signal $\hat{g}_r$ can only be equal to the original signal $\hat{g}$ if its periodical repetitions are not overlapping. This is exactly what the sampling theorem states. The Fourier transform of the ideal interpolation function is a box function which is one within the first Brillouin zone and zero outside, eliminating all replications and leaving the original band-limited signal $\hat{g}$ unchanged.

9.4 Quantization

9.4.1 Equidistant Quantization

After digitization (Section 9.2), the pixels still show continuous gray values. For use with a computer we must map them onto a limited number Q of discrete gray values:

$$[0, \infty[\xrightarrow{Q} \{g_0, g_1, \ldots, g_{Q-1}\} = G.$$

This process is called *quantization*, and we have already discussed some aspects thereof in Section 2.2.4. In this section, we discuss the errors related to quantization. Quantization always introduces errors, as the true value g is replaced by one of the quantization levels g_q. If the quantization levels are equally spaced with a distance Δg and if all gray values are equally probable, the variance introduced by the quantization is given by

$$\sigma_q^2 = \frac{1}{\Delta g} \int_{g_q - \Delta g/2}^{g_q + \Delta g/2} (g - g_q)^2 \mathrm{d}g = \frac{1}{12} (\Delta g)^2. \tag{9.24}$$

This equation shows how we select a quantization level. We take the level g_q for which the distance from the gray value g, $|g - g_q|$, is smaller than the neighboring quantization levels q_{k-1} and q_{k+1}. The standard deviation σ_q is about 0.3 times the distance between the quantization levels Δg.

Quantization with unevenly spaced quantization levels is hard to realize in any image processing system. An easier way to yield unevenly spaced levels is to use equally spaced quantization but to transform the intensity signal before quantization with a nonlinear amplifier, e. g., a logarithmic amplifier. In case of a logarithmic amplifier we would obtain levels whose widths increase proportionally with the gray value.

9.4.2 Accuracy of Quantized Gray Values

With respect to the quantization, the question arises of the accuracy with which we can measure a gray value. At first glance, the answer to this question seems to be trivial and given by Eq. (9.24): the maximum error is half a quantization level and the mean error is about 0.3 quantization levels.

But what if we measure the value repeatedly? This could happen if we take many images of the same object or if we have an object of a constant gray value and want to measure the mean gray value of the object by averaging over many pixels.

From the laws of statistical error propagation (Section 3.3.3), we know that the error of the mean value decreases with the number of measurements according to

$$\sigma_{\text{mean}} \approx \frac{1}{\sqrt{N}}\sigma, \tag{9.25}$$

where σ is the standard deviation of the individual measurements and N the number of measurements taken. This equation tells us that if we take 100 measurements, the error of the mean should be just about 1/10 of the error of the individual measurements.

Does this law apply to our case? Yes and no — it depends, and the answer appears to be a paradox. If we measure with a perfect system, i. e., without any noise, we would always get the same quantized value and, therefore, the result could not be more accurate than the individual measurements. However, if the measurements are noisy, we would obtain different values for each measurement. The probability for the different values reflects the mean and variance of the noisy signal, and because we can measure the distribution, we can estimate both the mean and the variance.

As an example, we take a standard deviation of the noise equal to the quantization level. Then, the standard deviation of an individual measurement is about 3 times larger than the standard deviation due to the quantization. However, already with 100 measurements, the standard deviation of the mean value is only 0.1, or 3 times lower than that of the quantization.

As in images we can easily obtain many measurements by spatial averaging, there is the potential for much more accurate mean values than given by the quantization level.

The accuracy is also limited, however, by other, systematic errors. The most significant source is the unevenness of the quantization levels. In a real quantizer, such as an analog to digital converter, the quantization levels are not equally distant but show systematic deviations which may be up to half a quantization interval. Thus, a careful investigation of the analog to digital converter is required to estimate what really limits the accuracy of the gray value measurements.

9.5 Further Readings‡

Sampling theory is detailed in Poularikas [141, Section 1.6]. A detailed account on sampling of random fields — also with random distances is given by Papoulis [134, Section 11.5]. Section 9.4 discusses only quantization with even bins. Quantization with uneven bins is expounded in Rosenfeld and Kak [157].

10 Pixel Processing

10.1 Introduction

After a digital image has been captured, the first preprocessing steps include two classes of operations, point operations and geometric operations. Essentially, these two types of operations modify the "what" and "where" of a pixel.

Point operations modify the gray values at individual pixels depending only on the gray value and possibly on the position of the pixels. Generally, such a kind of operation is expressed by

$$G'_{mn} = P_{mn}(G_{mn}). \tag{10.1}$$

The indices at the function P denote the possible dependency of the point operation on the position of the pixel.

In contrast, *geometric operations* modify only the position of a pixel. A pixel located at the position $\boldsymbol{x}$ is relocated to a new position $\boldsymbol{x}'$. The relation between the two coordinates is given by the geometric mapping function.

$$\boldsymbol{x}' = M(\boldsymbol{x}). \tag{10.2}$$

Point and geometric operations are complementary operations. They are useful for corrections of elementary distortions of the image formation process such as nonlinear and inhomogeneous radiometric responsivity of the imaging sensors or geometric distortions of the imaging system. We apply point operations to correct and optimize the image illumination, to detect underflow and overflow, to enhance and stretch contrast, to average images, to correct for inhomogeneous illumination, or to perform radiometric calibration (Sections 10.2.3–10.3.3).

Geometric operations include two major steps. In most applications, the mapping function Eq. (10.2) is not given explicitly but must be derived from the correspondences between the original object and its image (Section 10.5.4). When an image is warped by a geometric transform, the pixels in the original and warped images almost never fall onto each other. Thus, it is required to interpolate gray values at these pixels from neighboring pixels. This important task is discussed in detail in Section 10.6 because it is not trivial to perform accurate interpolation.

Point operations and geometric operations are not only of interest for elementary preprocessing steps. They are also an integral part of

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2

Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

many complex image operations, especially for feature extraction (Chapters 11–15). Note, however, that point operations and geometric operations are not suitable to correct the effects of an optical system described by its point spread function. This requires sophisticated reconstruction techniques that are discussed in Chapter 17. Point operations and geometric operations are limited to the performance of simple radiometric and geometric corrections.

10.2 Homogeneous Point Operations

10.2.1 Definitions and Basic Properties

If a point operation is independent of the position of the pixel, we call it a *homogeneous point operation* and write

$$G'_{mn} = P(G_{mn}). \tag{10.3}$$

A point operation maps the set of gray values onto itself. Generally, point operations are not invertible, as two different gray values may be mapped onto one. Thus, a point operation generally results in an irrecoverable loss of information. The point operation

$$P(q) = \begin{cases} 0 & q < t \\ Q-1 & q \geq t \end{cases}, \tag{10.4}$$

for example, performs a simple global threshold operation. All gray values below the threshold t are set to zero (black), all above and equal to the threshold to 255 (white). Consequently, this point operation cannot be inverted. An example for an invertible point operation is image negation. This operation computes an image with an inverted gray scale as with a photographic negative according to

$$P_N(q) = Q - 1 - q. \tag{10.5}$$

The inverse operation of a negation is another negation:

$$P_N\left(P_N(q)\right) = Q - 1 - (Q - 1 - q) = q. \tag{10.6}$$

Another example of an invertible point operation is the conversion between signed and unsigned representations of gray values (Section 2.2.5).

10.2.2 Look-up Tables

The direct computation of homogeneous point operations according to Eq. (10.3) may be very costly. This is demonstrated by the following example. The 14-bit gray scale of a 1024×1024 image from a high-resolution CCD camera is to be presented in an 8-bit logarithmic gray

scale covering 4.3 decades from 1 to 16 383. The following point operation performs this conversion:

$$P(q) = 59.30 \lg q \tag{10.7}$$

A straightforward implementation would require the following operations per pixel: an integer to double conversion, computation of the logarithm, multiplication by 59.30, and a double to 8-bit integer conversion. All these operations have to be repeated over a million times for a 1024×1024 image.

The key point for a more efficient implementation lies in the observation that the definition range of any point operation consists of only a limited number of Q quantization levels. For the 14-bit to 8-bit logarithmic conversion, we have at most 16 384 different input values. This means that most of the one million computations are just repeated, on average 64 times. We can avoid the unnecessary repetition by precalculating $P(q)$ for all 16 384 possible gray values and storing the computed values in a 16 384-element table. Then, the computation of the point operation is reduced to a replacement of the gray value by the element in the table with an index corresponding to the gray value.

Such a table is called a *look-up table* or *LUT*. Hence, homogeneous point operations are equivalent to *look-up table operations*. Look-up tables are more efficient the smaller the number of quantization levels. For standard 8-bit images, the tables contain just 256 values. But it is still efficient in most cases to use 65 536 entry look-up tables with 16-bit images.

In most image processing systems and frame grabbers, look-up tables are implemented in hardware. There are two possible places for look-up tables on frame grabber boards, as illustrated in Fig. 10.1. The *input LUT* is located between the *analog-digital converter* and the frame buffer. The *output LUT* is located between the frame buffer and the digital-analog converter for output of the image in the form of an analog video signal, e. g., to a monitor. The input LUT allows a point operation to be performed *before* the image is stored in the frame buffer. With the output LUT, a point operation can be performed and observed on the monitor. In this way, we can interactively perform point operations *without* modifying the stored image. Many modern frame grabbers no longer include a frame buffer. With the advent of fast peripheral bus systems (such as the PCI bus with a peak rate of 132 MB/s, see Section 1.7), digitized images can be transferred directly to the PC memory (Fig. 10.2). With such a frame grabber, image display is performed on the graphics board of the computer. Consequently, the frame grabber includes only an input look-up table.

The use of input LUTs is limited. Nonlinear LUT functions lead to missing gray values or map two consecutive values onto one (Fig. 10.3).

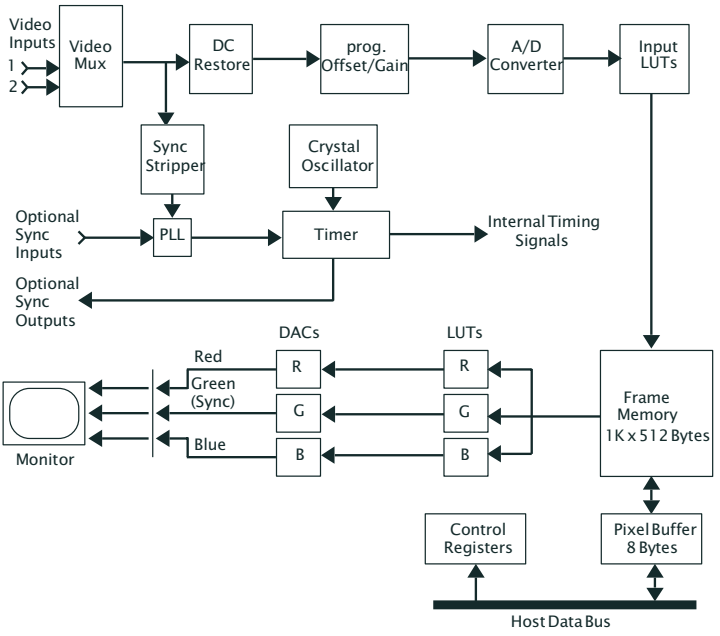

Figure 10.1: *Block diagram of the PCVISIONplus frame grabber from Imaging Technology, Inc. Look-up tables are located between the A/D converter and frame buffer (input LUT) and the frame buffer and display (output LUT).*

In this way, artifacts are introduced that yield enhanced errors in subsequent processing such as the computation of mean values and edge detection. It is obvious that especially the steepness of edges and the accuracy of gray value changes are affected.

Input LUTs would be valuable also for nonlinear point operations if the 8-bit input values were mapped to higher precision output values, e. g., 16-bit integers or 32-bit floating point numbers. Then rounding errors could be avoided. At the same time, the gray levels could be converted into a calibrated signal, e. g., a temperature for an infrared camera. Unfortunately, such generalized LUTs are not yet implemented in hardware. However, it is easy to realize them in software.

In contrast to the input LUT, the output LUT is a much more widely used tool, as it does not change the stored image. With LUT operations, we can also convert a gray value image into a *pseudo-color image*. Again, this technique is common even with the simplest frame grabber boards (Fig. 10.1). Not much additional hardware is needed. Three digital-analog converters are used for the primary colors red, green, and blue. Each channel has its own LUT with 256 entries for an 8-bit display. In this way, we can map each individual gray value q to any color by assigning a color triple to the corresponding LUT addresses $r(q)$, $g(q)$, and $b(q)$.

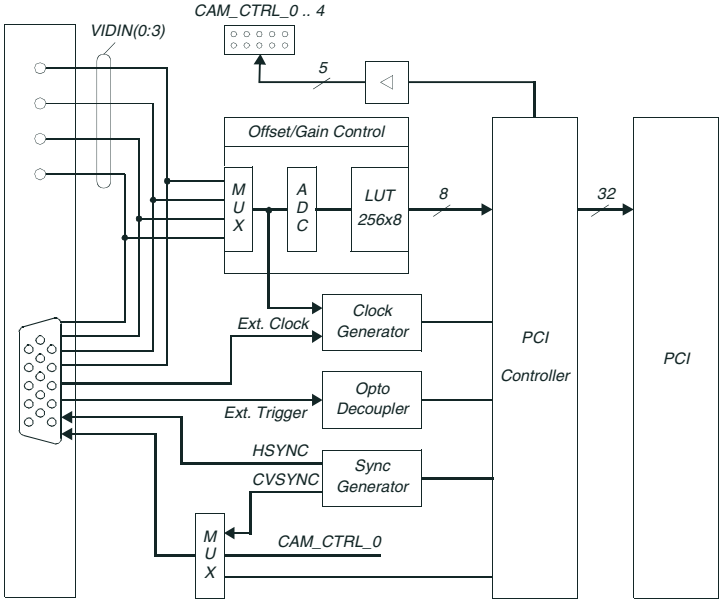

Figure 10.2: *Block diagram of the PCEYE_1 frame grabber from ELTEC Elektronik GmbH as an example of a modern PCI bus frame grabber without a frame buffer. The image data are transferred in realtime via direct memory access (DMA) to the memory of the PC for display and further processing.*

Formally, this is a *vector* point operation

$$\boldsymbol{P}(q) = \begin{bmatrix} r(q) \\ g(q) \\ b(q) \end{bmatrix}. \tag{10.8}$$

When all three point functions $r(q)$, $g(q)$, and $b(q)$ are identical, a gray tone will be displayed. If two of the point functions are zero, the image will appear in the remaining color.

10.2.3 Interactive Gray Value Evaluation

Homogeneous point operators implemented via look-up tables are a very useful tool for inspecting images. As the look-up table operations work in real-time, images can be manipulated interactively. If only the output look-up table is changed, the original image content remains unchanged. Here, we demonstrate typical tasks.

Evaluating and Optimizing Illumination. With the naked eye, we can hardly estimate the homogeneity of an illuminated area as demonstrated

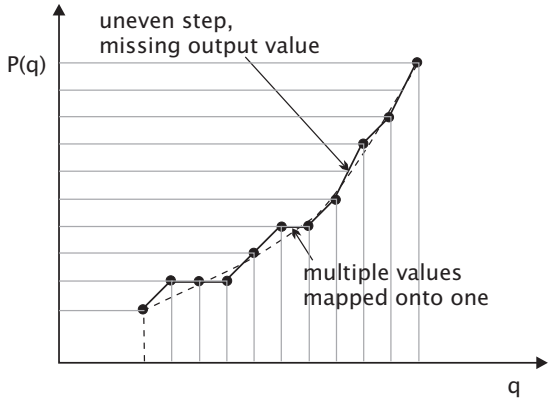

Figure 10.3: *Illustration of a nonlinear look-up table with mapping of multiple values onto one and missing output value leading to uneven steps.*

in Fig. 10.4a,b. A histogram reveals the gray scale distribution but not its spatial variation (Fig. 10.4c,d). Therefore, a histogram is not of much help for optimizing the illumination interactively. We need to mark gray scales such that absolute gray levels become perceivable for the human eye. If the radiance distribution is continuous, it is sufficient to use equidensities. This technique uses a staircase type of homogeneous point operation by mapping a certain range of gray scales onto one. This point operation is achieved by zeroing the p least significant bits with a logical *and* operation:

$$q' = P(q) = q \wedge \overline{(2^p - 1)}, \tag{10.9}$$

where $\wedge$ denotes the logical (bitwise) *and* and overlining denotes *negation*. This point operation limits the resolution to $Q - p$ bits and, thus, 2^{Q-p} quantization levels. Now, the jump between the remaining quantization levels is large enough to be perceived by the eye and we see contour lines of equal absolute gray scale in the image (Fig. 10.4). We can now try to homogenize the illumination by making the distance between the contour lines as large as possible.

Another way to mark absolute gray values is the so-called *pseudocolor image* that has already been discussed in Section 10.2.2. With this technique, a gray level q is mapped onto an RGB triple for display. As color is much better recognized by the eye, it helps reveal absolute gray levels.

Detection of Underflow and Overflow. Under- and overflows of the gray values of a digitized image often go unnoticed and cause a serious bias in further processing, for instance for mean gray values of objects or the center of gravity of an object. In most cases, such areas cannot

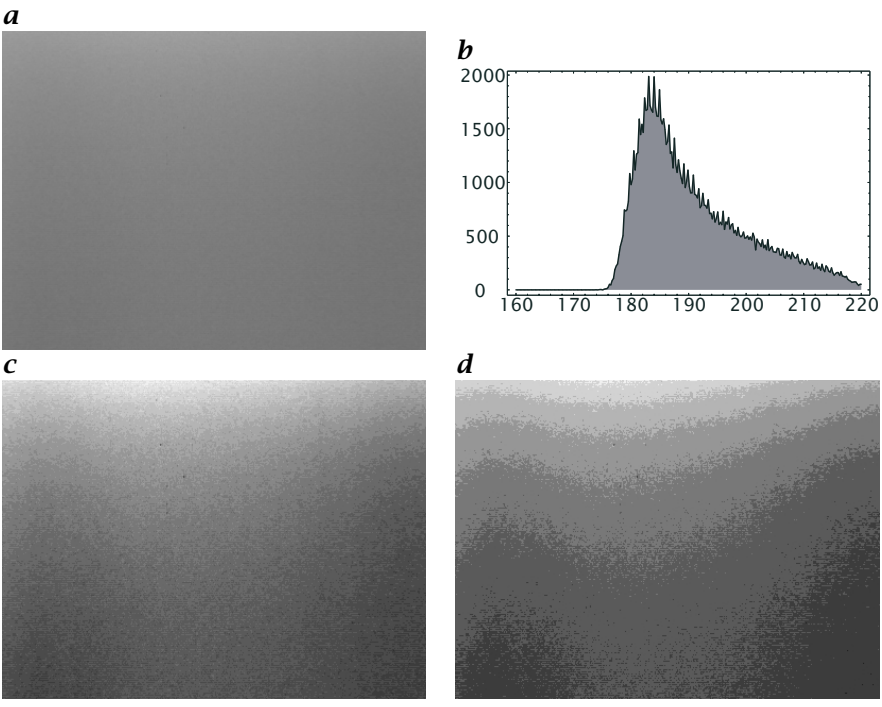

Figure 10.4: ***a*** *The irradiance is gradually decreasing from the top to the bottom, which is almost not recognized by the eye. The gray scale of this floating-point image computed by averaging over 100 images ranges from 160 to 200.* ***b*** *Histogram of* ***a****;* ***c*** *and* ***d*** *(contrast enhanced, gray scale 184–200): Edges artificially produced by a staircase LUT with a step height of 1.0 and 2.0 make contours of constant irradiance easily visible.*

be detected directly. They may only become apparent in textured areas when the texture is bleached out. Over and underflow are detected easily in histograms by strong peaks at the minimum and/or maximum gray values (Fig. 10.5). With pseudocolor mapping, the few lowest and highest gray values could be displayed, for example, in blue and red, respectively. Then, gray values dangerously close to the limits immediately pop out of the image and can be avoided by correcting the illumination lens aperture or gain of the video input circuit of the frame grabber.

Contrast enhancement. Because of poor illumination conditions, it often happens that images are underexposed. Then, the image is too dark and of low contrast (Fig. 10.6a). The histogram (Fig. 10.6b) shows that the image contains only a small range of gray values at low gray values. The appearance of the image improves considerably if we apply a point operation which maps a small grayscale range to the full contrast range

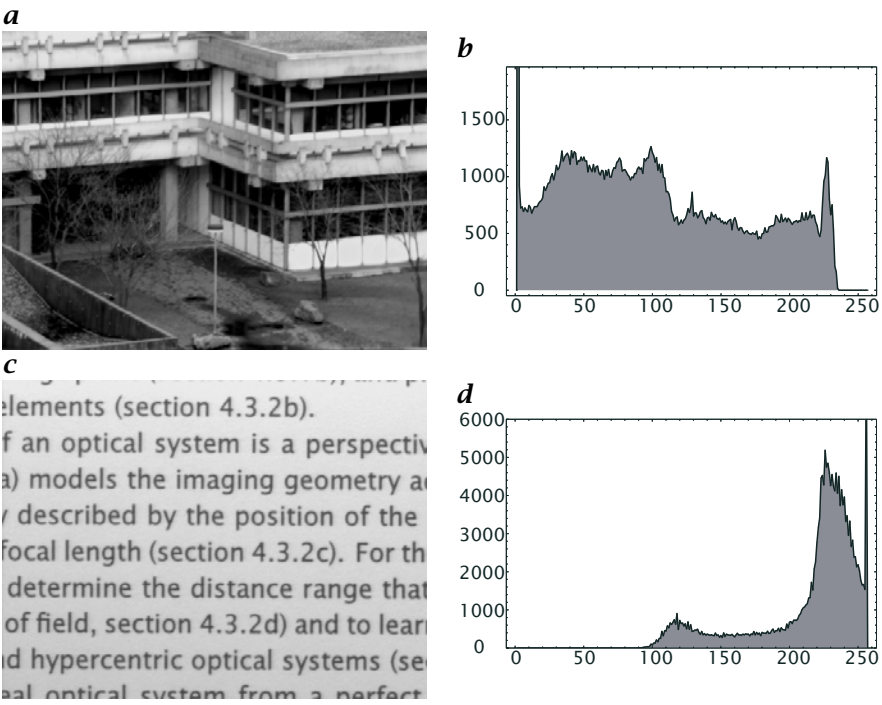

Figure 10.5: *Detection of underflow and overflow in digitized images by histograms;* ***a*** *image with underflow and* ***b*** *its histogram;* ***c*** *image with overflow and* ***d*** *its histogram.*

(for example with this operation: $q' = 4q$ for $q < 64$, and $q' = 255$ for $q \geq 64$) (Fig. 10.6c). We only improve the appearance of the image but not the image *quality* itself. The histogram shows that the gray value resolution is still the same (Fig. 10.6d).

The image quality can be improved. The best way is to increase the object irradiance by using a more powerful light source or a better design of the illumination setup. If this is not possible, we can still increase the gain of the analog video amplifier. All modern image processing boards include an amplifier whose gain and offset can be set by software (see Figs. 10.1 and 10.2). By increasing the gain, the brightness and resolution of the image improve, but at the expense of an increased noise level.

Contrast Stretching. It is often of interest to analyze faint irradiance differences which are beyond the resolution of the human visual system or the display equipment used. This is especially the case if images are printed. In order to observe faint differences, we stretch a small gray scale range of interest to the full range available. All gray values outside this range are set to the minimum or maximum value. This operation

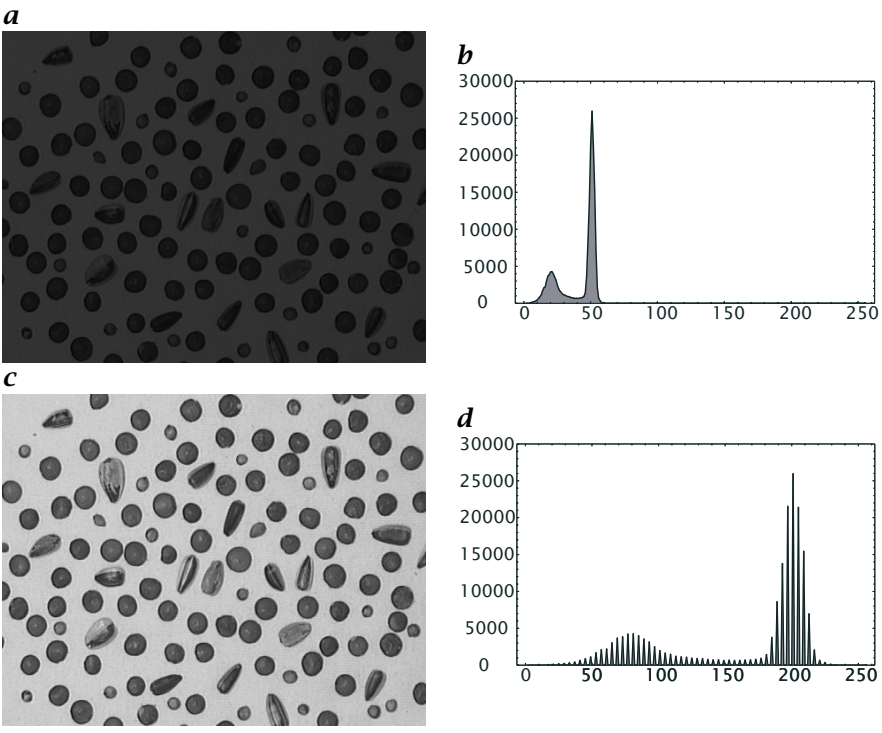

Figure 10.6: *Contrast enhancement;* ***a*** *underexposed image and* ***b*** *its histogram;* ***c*** *interactively contrast enhanced image and* ***d*** *its histogram.*

requires that the gray values of the object of interest fall into the range selected for contrast stretching. An example of contrast stretching is shown in Fig. 10.7a,b. The wedge at the bottom of the images, ranging from 0 to 255, directly shows which part of the gray scale range is contrast enhanced.

Range Compression. In comparison to the human visual system, a digital image has a considerably smaller dynamical range. If a minimum resolution of 10% is demanded, the gray values must not be lower than 10. Therefore, the maximum dynamical range in an 8-bit image is only $255/10 \approx 25$. The low contrast range of digital images makes them appear of low quality when high-contrast scenes are encountered. Either the bright parts are bleached or no details can be recognized in the dark parts. The dynamical range can be increased by a transform that was introduced in Section 2.2.6 as the *gamma transform*. This nonlinear homogeneous point operation has the form

$$q' = \frac{255}{255^{\gamma}} q^{\gamma}. \tag{10.10}$$

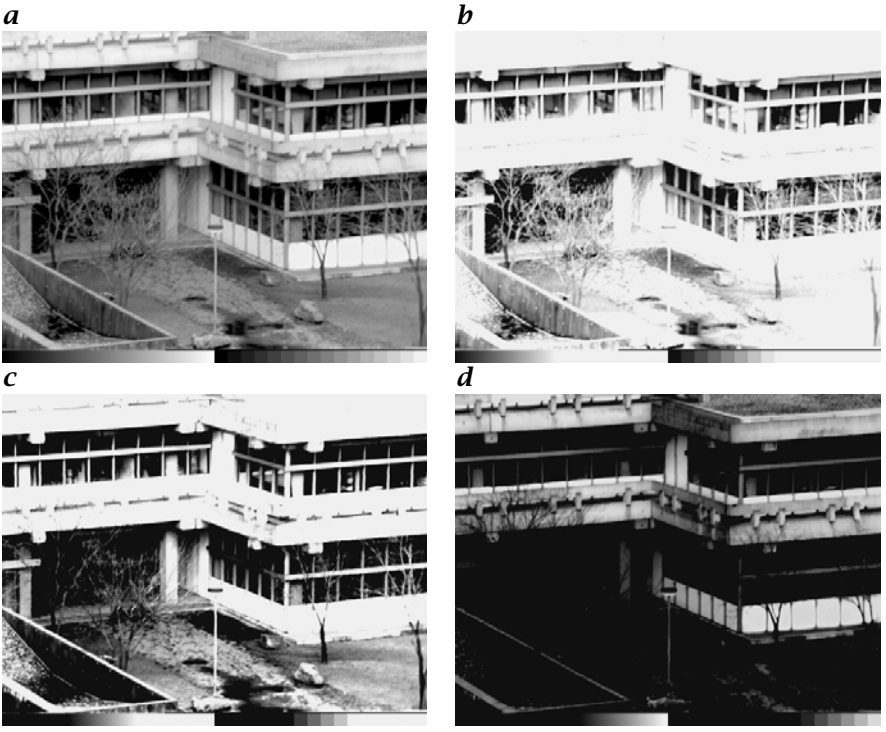

Figure 10.7: b–d *Contrast stretching of the image shown in* ***a****. The stretched range can be read from the transformation of the gray scale wedge at the bottom of the image.*

The factors in Eq. (10.10) are chosen such that a range of $[0, 255]$ is mapped onto itself. This transformation allows a larger dynamic range to be recognized at the cost of resolution in the bright parts of the image. The dark parts become brighter and show more details. This contrast transformation is better adapted to the logarithmic characteristics of the human visual system. An image presented with different gamma factors is shown in Fig. 10.8.

Noise Variance Equalization. From Section 3.4.5, we know that the variance of the noise generally depends on the image intensity according to

$$\sigma_g^2(g) = \sigma_0^2 + \alpha g. \tag{10.11}$$

A statistical analysis of images and image operations is, however, much easier if the noise is independent of the gray value. Only then all the error propagation techniques discussed in Section 3.3.3 are valid.

Thus we need to apply a nonlinear gray value transform $h(g)$ in such a way that the noise variance becomes constant. In first order, the vari-

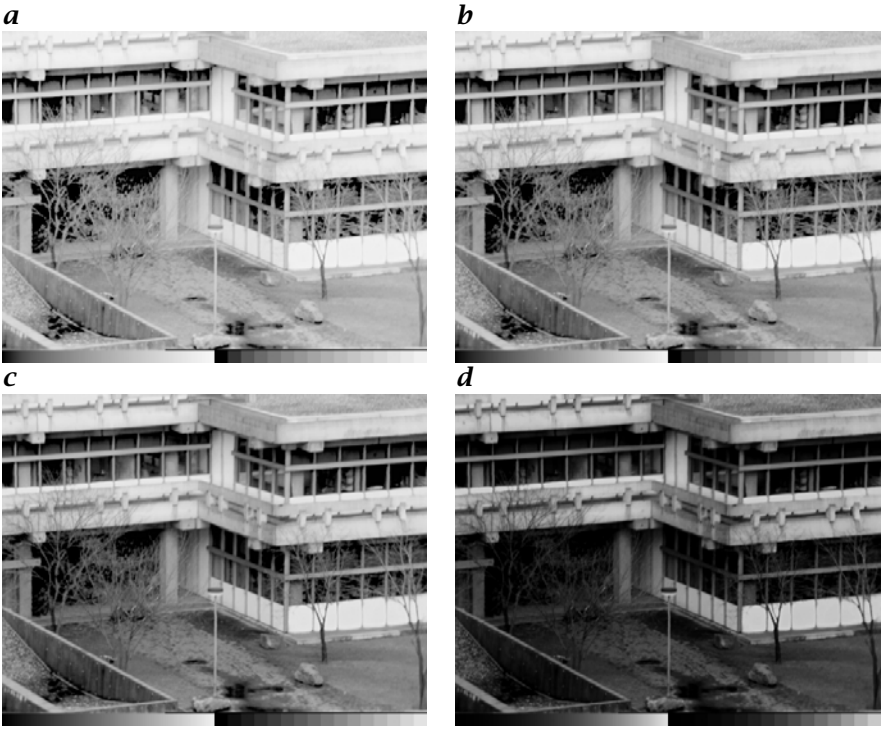

Figure 10.8: *Presentation of an image with different gamma values:* ***a*** *0.5,* ***b*** *0.7,* ***c*** *1.0, and* ***d*** *2.0.*

ance of $h(g)$ is

$$\sigma_h^2 \approx \left(\frac{\mathrm{d}h}{\mathrm{d}g}\right)^2 \sigma_g^2(g) \tag{10.12}$$

according to Eq. (3.35) [47]. If we set σ_h^2 to be constant, we obtain

$$\mathrm{d}h = \frac{\sigma_h}{\sqrt{\sigma^2(g)}}\mathrm{d}g.$$

Integration yields

$$h(g) = \sigma_h \int_0^g \frac{\mathrm{d}g'}{\sqrt{\sigma^2(g')}} + C. \tag{10.13}$$

The two free parameters σ_h and C can, for instance, be used to fit the values of h into a suitable interval. With the linear variance function Eq. (10.11), the integral in Eq. (10.13) yields

$$h(g) = \frac{2\sigma_h}{\sqrt{\alpha}}\sqrt{\sigma_0^2 + \alpha g} + C. \tag{10.14}$$

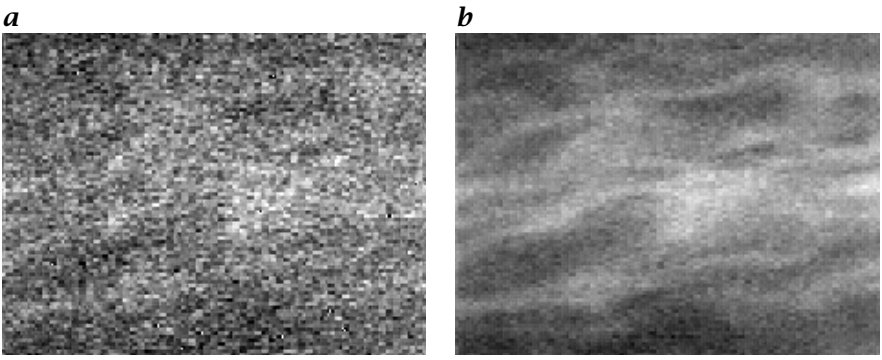

Figure 10.9: *Noise reduction by image averaging:* ***a*** *single thermal image of small temperature fluctuations on the water surface cooled by evaporation;* ***b*** *same, averaged over 16 images; the full gray value range corresponds to a temperature range of 1.1 K.*

The nonlinear transform becomes particularly simple for an ideal imaging sensor with $\sigma_0 = 0$. Then a square root transform must be applied to obtain an intensity independent noise variance:

$$h(g) = \frac{2\sigma_h}{\sqrt{\alpha}}\sqrt{g}. \tag{10.15}$$

10.3 Inhomogeneous Point Operations†

Homogeneous point operations are only a subclass of point operators. In general, a point operation depends also on the position of the pixel in the image. Such an operation is called an *inhomogeneous point operation*. Inhomogeneous point operations are mostly related to calibration procedures. Generally, the computation of an inhomogeneous point operation is much more time consuming than the computation of a homogeneous point operation. We cannot use look-up tables since the point operation depends on the pixel position and we are forced to calculate the function for each pixel.

The subtraction of a background image without objects or illumination is a simple example of an inhomogeneous point operation which is written as:

$$G'_{mn} = P_{mn}(G_{mn}) = G_{mn} - B_{mn}, \tag{10.16}$$

where B_{mn} is the background image.

10.3.1 Image Averaging

One of the simplest inhomogeneous point operations is *image averaging*. There are a number of imaging sensors available which show a

considerable noise level. Prominent examples include *thermal imaging* (Section 6.4.1) and all sensor types such as slow-scan CCD imagers or image amplifiers where only a limited number of photons are collected.

Figure 10.9a shows the temperature differences at the water surface of a wind-wave facility cooled at 1.8 m/s wind speed by evaporation. Because of a substantial noise level, the small temperature fluctuations can hardly be detected. Taking the mean over several images significantly reduces the noise level (Fig. 10.9b).

The error of the mean (Section 3.3.3) taken from N samples is given by

$$\sigma_{\overline{g}}^2 \approx \frac{1}{(N-1)}\sigma_g^2 = \frac{1}{N(N-1)}\sum_{n=0}^{N}(g-\overline{g})^2. \tag{10.17}$$

If we take the average of N images, the noise level is reduced by $1/\sqrt{N}$ compared to a single image. Taking the mean over 16 images thus reduces the noise level by a factor of four. Equation (10.17) is only valid, however, if the standard deviation σ_g is significantly larger than the standard deviation related to the quantization (Section 9.4).

10.3.2 Correction of Inhomogeneous Illumination

Every real-world application has to contend with *uneven illumination* of the observed scene. Even if we spend a lot of effort optimizing the illumination setup, it is still very hard to obtain a perfectly even object irradiance. A nasty problem is caused by small dust particles in the optical path, especially on the glass window close to the CCD sensor. Because of the distance of the window from the imager, these particles — if they are not too large — are blurred to such an extent that they are not directly visible. But they still absorb some light and thus cause a drop in the illumination level in a small area. These effects are not easily visible in a scene with high contrast and many details, but become very apparent in the case of a uniform background (Fig. 10.4a and b). CCD sensors also show an uneven sensitivity of the individual photoreceptors which adds to the nonuniformity of the image. These distortions severely limit the quality of the images. These effects make it more difficult to separate an object from the background, and introduce systematic errors for subsequent image processing steps.

Nevertheless, it is possible to correct these effects if we know the nature of the distortion and can take suitable reference images. In the following, we study two cases. In the first, we assume that the gray value in the image is a product of the inhomogeneous irradiance and the reflectivity or transmissivity of the object. Furthermore, we assume that we can take a reference image without absorbing objects or with an object of constant reflectivity. A reference image can also be computed, when

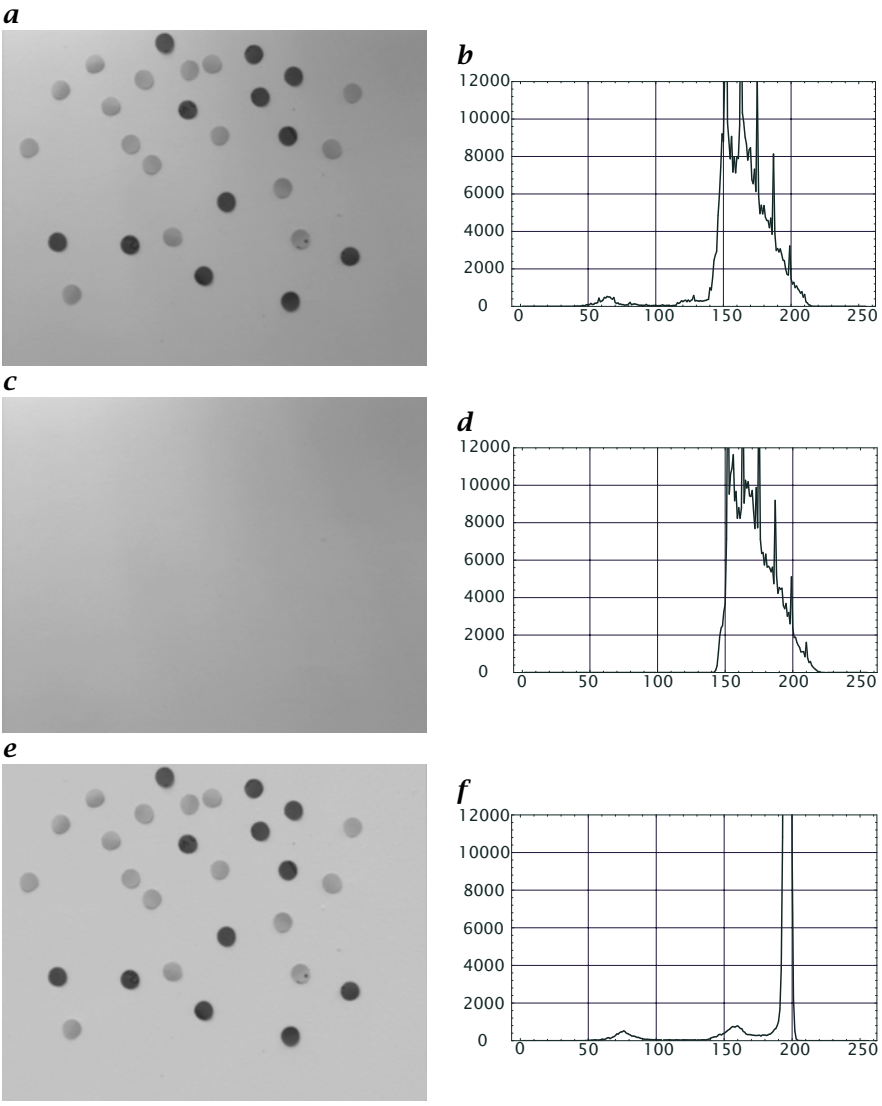

Figure 10.10: *Correction of uneven illumination with an inhomogeneous point operation:* ***a*** *original image and* ***b*** *its histogram;* ***c*** *background image and* ***d*** *its histogram;* ***e*** *division of the image by the background image and* ***f*** *its histogram.*

small objects are randomly distributed in the image. Then, it is sufficient to compute the average image from many images with the objects. The inhomogeneous illumination can then be corrected by dividing the image by the reference image:

$$\boldsymbol{G}' = c \cdot \boldsymbol{G}/\boldsymbol{R}. \tag{10.18}$$

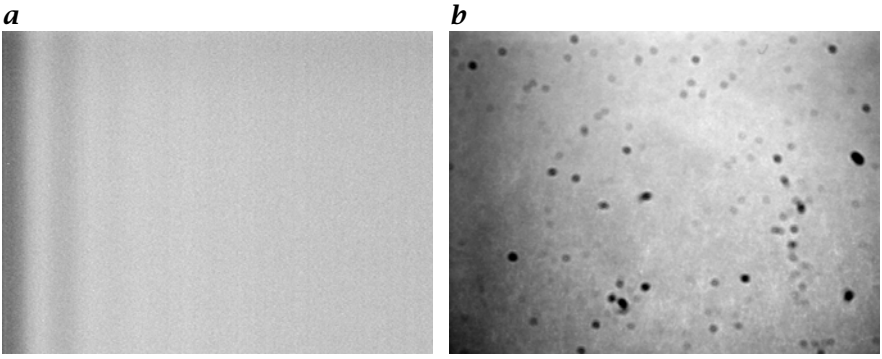

Figure 10.11: *Contrast-enhanced* ***a*** *dark image and* ***b*** *reference image for a two-point radiometric calibration of a CCD camera with analog video output.*

The constant c is required to represent the normalized image with integer numbers again. If the objects absorb light, the constant c is normally chosen to be close to the maximum integer value. Figure 10.10e demonstrates that an effective suppression of inhomogeneous illumination is possible using this simple method.

10.3.3 Two-Point Radiometric Calibration

The simple ratio imaging described above is not applicable if also a non-zero inhomogeneous background has to be corrected for, as caused, for instance, by the fixed pattern noise of a CCD sensor. In this case, two reference images are required. This technique is also applied for a simple *two-point radiometric calibration* of an imaging sensor with a linear response. Some image measuring tasks require an absolute or relative radiometric calibration. Once such a calibration is obtained, we can infer the radiance of the objects from the measured gray values.

First, we take a dark image $\boldsymbol{B}$ without any illumination. Second, we take a reference image $\boldsymbol{R}$ with an object of constant radiance, e. g., by looking into an *integrating sphere*. Then, a normalized image corrected for both the fixed pattern noise and inhomogeneous sensitivity is given by

$$\boldsymbol{G}' = c\frac{\boldsymbol{G}-\boldsymbol{B}}{\boldsymbol{R}-\boldsymbol{B}}. \tag{10.19}$$

Fig. 10.11 shows a contrast-enhanced dark image and reference image of a CCD camera with analog output. Typical signal distortions can be observed. The signal oscillation at the left edge of the dark image results from an electronic interference, while the dark blobs in the reference image are caused by dust on the glass window in front of the sensor. The improvement due to the radiometric calibration can clearly be seen in Fig. 10.12.

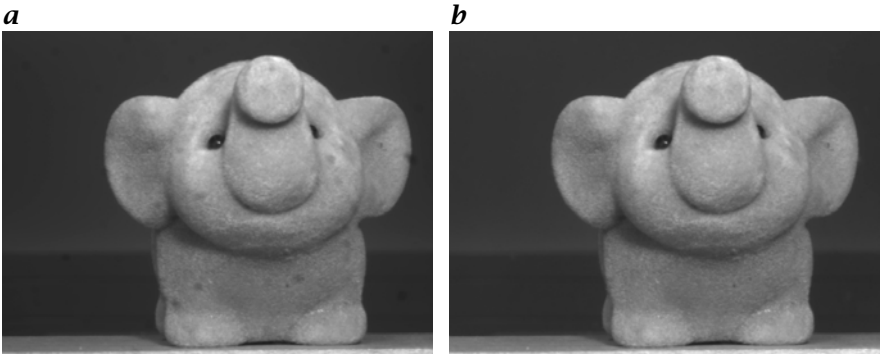

Figure 10.12: *Two-point radiometric calibration with the dark and reference image from Fig. 10.11:* ***a*** *original image and* ***b*** *calibrated image; in the calibrated image the dark spots caused by dust are no longer visible.*

10.3.4 Nonlinear Radiometric Calibration‡

Sometimes, the quantity to be measured by an imaging sensor is related in a nonlinear way to the measured gray value. An obvious example is thermography. In such cases a *nonlinear radiometric calibration* is required. Here, the temperature of the emitted object is determined from its radiance using Planck's equations (Section 6.4.1).

We will give a practical calibration procedure for ambient temperatures. Because of the nonlinear relation between radiance and temperature, a simple two-point calibration with linear interpolation is not sufficient. Haußecker [63] showed that a quadratic relation is accurate enough for a small temperature range, say from 0 to 40° centigrade. Therefore, three calibration temperatures are required, which are provided by a temperature-regulated blackbody calibration unit.

The calibration delivers three calibration images $\boldsymbol{G}_1$, $\boldsymbol{G}_2$, and $\boldsymbol{G}_3$ with known temperatures T_1, T_2, and T_3. The temperature image $\boldsymbol{T}$ of an arbitrary image $\boldsymbol{G}$ can be computed by quadratic interpolation as

$$\boldsymbol{T} = \frac{\Delta\boldsymbol{G}_2 \cdot \Delta\boldsymbol{G}_3}{\Delta\boldsymbol{G}_{21} \cdot \Delta\boldsymbol{G}_{31}} T_1 - \frac{\Delta\boldsymbol{G}_1 \cdot \Delta\boldsymbol{G}_3}{\Delta\boldsymbol{G}_{21} \cdot \Delta\boldsymbol{G}_{32}} T_2 + \frac{\Delta\boldsymbol{G}_1 \cdot \Delta\boldsymbol{G}_2}{\Delta\boldsymbol{G}_{31} \cdot \Delta\boldsymbol{G}_{32}} T_3, \tag{10.20}$$

with

$$\Delta\boldsymbol{G}_k = \boldsymbol{G} - \boldsymbol{G}_k \qquad \text{and} \qquad \Delta\boldsymbol{G}_{kl} = \boldsymbol{G}_k - \boldsymbol{G}_l. \tag{10.21}$$

The symbol $\cdot$ indicates pointwise multiplication of the images in order to distinguish it from matrix multiplication. Figure 10.13a, b, and c shows three calibration images. The infrared camera looks at the calibration target via a mirror which limits the field of view at the edges of the images. This is the reason for the sharp temperature changes seen at the image borders in Fig. 10.13a, c. The calibration procedure removes the residual inhomogeneities (Fig. 10.13d, f) that show up in the original images.

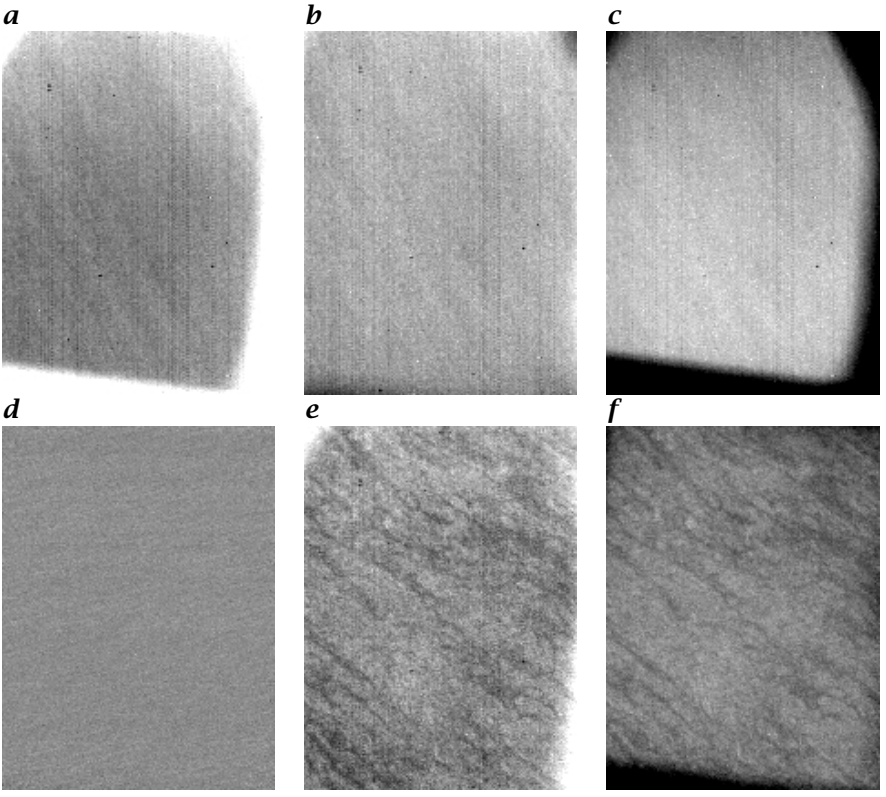

Figure 10.13: *Three-point calibration of infrared temperature images:* ***a–c*** *show images of calibration targets made out of aluminum blocks at temperatures of 13.06, 17.62, and 22.28°centigrade. The images are stretched in contrast to a narrow range of the 12-bit digital output range of the infrared camera:* ***a****: 1715–1740,* ***b****: 1925–1950,* ***c****: 2200–2230, and show some residual inhomogeneities, especially vertical stripes.* ***d*** *Calibrated image using the three images* ***a–c*** *with quadratic interpolation.* ***e*** *Original and* ***f*** *calibrated image of the temperature microscale fluctuations at the ocean surface (area about* $0.8 \times 1.0\ m^2$*).*

10.3.5 Windowing

Another important application of inhomogeneous point operations is an operation known as *windowing*. Before we can calculate the DFT of an image, the image must be multiplied with a *window function*. If we omit this step, the spectrum will be distorted by the convolution of the image spectrum with the Fourier transform of the box function, the sinc function (see Section 2.3, ➢R5), which causes spectral peaks to become star-like patterns along the coordinate axes in Fourier space (Fig. 10.14b). We can also explain these distortions with the periodic repetition of finite-area images, an effect that is discussed in conjunc-

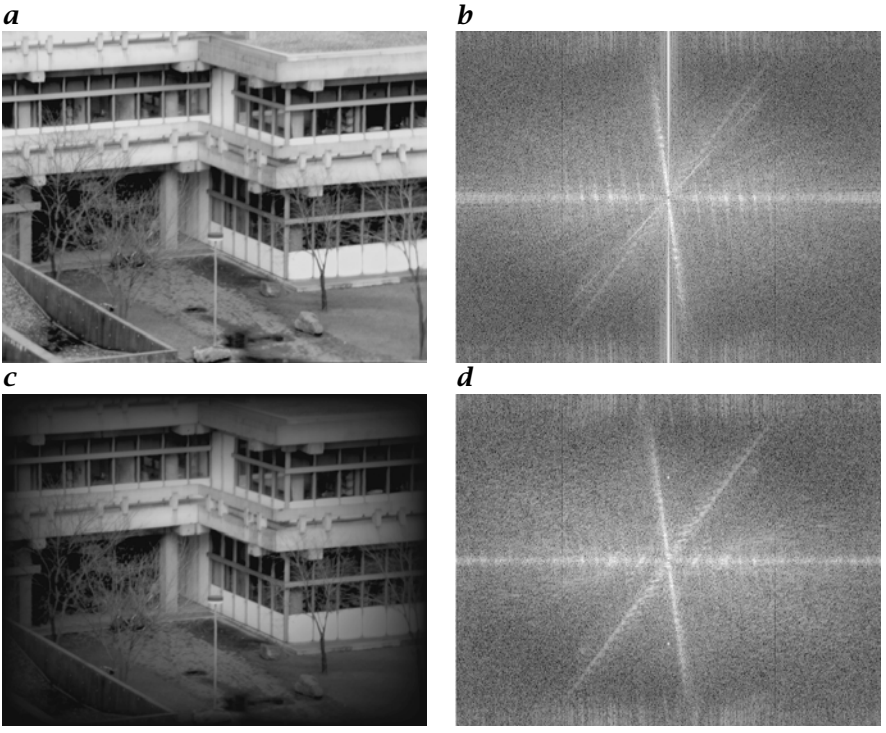

Figure 10.14: *Effect of windowing on the discrete Fourier transform:* ***a*** *original image;* ***b*** *DFT of* ***a*** *without using a window function;* ***c*** *image multiplied with a cosine window;* ***d*** *DFT of* ***c*** *using a cosine window.*

tion with the sampling theorem in Section 9.2.3. The periodic repetition leads to discontinuities at the horizontal and vertical edges of the image which cause correspondingly high spectral densities along the x and y axes in the Fourier domain.

In order to avoid these distortions, we must multiply the image with a window function that gradually approaches zero towards the edges of the image. An optimum window function should preserve a high spectral resolution and show minimum distortions in the spectrum, that is, its DFT should fall off as fast as possible. These are contradictory requirements. A good spectral resolution requires a broad window function. Such a window, however, falls off steeply at the edges, causing a slow fall-off of the sidelobes of its spectrum.

A carefully chosen window is crucial for a spectral analysis of time series [119, 133]. However, in digital image processing it is less critical because of the much lower dynamic range of the gray values. A simple

cosine window

$$W_{mn} = \sin\left(\frac{\pi m}{M}\right)\sin\left(\frac{\pi n}{N}\right), \quad 0 \le m < M,\ 0 \le n < N \tag{10.22}$$

performs this task well (Fig. 10.14c,d).

A direct implementation of the windowing operation is very time consuming because we would have to calculate the cosine function $2MN$ times. It is much more efficient to perform the calculation of the window function once, to store the window image, and to use it then for the calculation of many DFTs. The storage requirements can be reduced by recognizing that the window function Eq. (10.22) is separable, i. e., a product of two functions $W_{m,n} = {}^{c}w_m \cdot {}^{r}w_n$. Then, we need to calculate only the M plus N values for the column and row functions ${}^{c}w_m$ and ${}^{r}w_n$, respectively. As a result, it is sufficient to store only the row and column functions. The reduced storage space comes at the expense of an additional multiplication per pixel for the window operation.

10.4 Multichannel Point Operations‡

10.4.1 Definitions‡

Point operations can be generalized to multichannel point operations in a straightforward way. The operation still depends only on the values of a single pixel. The only difference is that it depends on a vectorial input instead of a scalar input. Likewise, the output image can be a multichannel image. For homogeneous point operations that do not depend on the position of the pixel in the image, we can write

$$\boldsymbol{G}' = \boldsymbol{P}(\boldsymbol{G}) \tag{10.23}$$

with

$$\begin{aligned} \boldsymbol{G}' &= [\boldsymbol{G}'_0\ \boldsymbol{G}'_1\ \ldots\ \boldsymbol{G}'_l\ \ldots\ \boldsymbol{G}'_{L-1}] \\ \boldsymbol{G} &= [\boldsymbol{G}_0\ \boldsymbol{G}_1\ \ldots\ \boldsymbol{G}_k\ \ldots\ \boldsymbol{G}_{K-1}], \end{aligned} \tag{10.24}$$

where $\boldsymbol{G}'_l$ and $\boldsymbol{G}_k$ are the components l and k of the multichannel images $\boldsymbol{G}'$ and $\boldsymbol{G}$ with L and K channels, respectively.

An important subclass of multicomponent point operators are linear operations. This means that each component of the multichannel image $\boldsymbol{G}'$ is a linear combination of the components of the multichannel image $\boldsymbol{G}$:

$$\boldsymbol{G}'_l = \sum_{k=0}^{K-1} P_{lk}\boldsymbol{G}_k \tag{10.25}$$

where P_{lk} are constant coefficients. Therefore, a general linear multicomponent point operation is given by a matrix (or tensor) of coefficients P_{lk}. Then, we can write Eq. (10.25) in matrix notation as

$$\boldsymbol{G}' = \boldsymbol{P}\boldsymbol{G} \tag{10.26}$$

where $\boldsymbol{P}$ is the matrix of coefficients.

If the components of the multichannel images in a point operation are not interrelated to each other, all coefficients in $\boldsymbol{P}$ except those on the diagonal become zero. For K-channel input and output images, just K different point operations remain, one for each channel. The matrix of point operations finally reduces to a standard scalar point operation when the same point operation is applied to each channel of a multi-component image.

For an equal number of output and input images, linear point operations can be interpreted as coordinate transformation. If the matrix of the coefficients in Eq. (10.26) has a rank $R < K$, the multichannel point operation projects the K-dimensional space to an R-dimensional subspace.

Generally, linear multichannel point operations are quite easy to handle as they can be described in a straightforward way with the concepts of linear algebra. For square matrices, for instance, we can easily give the condition when an inverse operation to a multichannel operation exists and compute it. For nonlinear multicomponent point operations, the linear coefficients in Eqs. (10.25) and (10.26) have to be replaced by nonlinear functions:

$$\boldsymbol{G}'_l = P_l(\boldsymbol{G}_0, \boldsymbol{G}_1, \ldots, \boldsymbol{G}_{K-1}) \tag{10.27}$$

Nonlinear multicomponent point operations cannot be handled in a general way, unlike linear operations. Thus, they must be considered individually. The complexity can be reduced significantly if it is possible to separate a given multichannel point operation into its linear and nonlinear parts.

10.4.2 Dyadic Point Operations‡

Operations in which only two images are involved are termed *dyadic point operations*. Dyadic homogeneous point operations can be implemented as LUT operations. Generally, any dyadic image operation can be expressed as

$$G'_{mn} = P(G_{mn}, H_{mn}). \tag{10.28}$$

If the gray values of the two input images take Q different values, there are Q^2 combinations of input parameters and, thus, different output values. Thus, for 8-bit images, 64K values need to be calculated. This is still a quarter less then with a direct computation for each pixel in a 512×512 image. All possible results of the dyadic operation can be stored in a large LUT L with $Q^2 = 64\text{K}$ entries in the following manner:

$$L(2^8 p + q) = P(p, q), \quad 0 \le p, q < Q. \tag{10.29}$$

The high and low bytes of the LUT address are given by the gray values in the images G and H, respectively.

Some image processing systems contain a 16-bit LUT as a modular processing element. Computation of a dyadic point operation either with a hardware or software LUT is often significantly faster than a direct implementation, especially if the operation is complex. In addition, it is easier to control exceptions such as division by zero or underflow and overflow.

A dyadic point operation can be used to perform two point operations simultaneously. The phase and magnitude (r, i) of a complex-valued image, for example, can be computed simultaneously with one dyadic LUT operation if we

restrict the output to 8 bits as well:

$$L(2^8 r + i) = 2^8 \sqrt{r^2 + i^2} + \frac{128}{\pi} \arctan\left(\frac{i}{r}\right), \quad 0 \le r, i < Q. \tag{10.30}$$

The magnitude is returned in the high byte and the phase, scaled to the interval $[-128, 127]$, in the low byte.

10.5 Geometric Transformations

In the remaining part of this chapter, we discuss geometric operations as the complementary operations to point operations. First we discuss elementary geometric transforms such as the affine transform (Section 10.5.2), the perspective transform (Section 10.5.3), and how to obtain the transformation parameters by point matching methods. Then we focus in Section 10.6 on interpolation, which arises as the major problem for a fast and accurate implementation of geometric operations on discrete images. Finally, in Section 10.6.7 we briefly discuss fast algorithms for geometric transforms.

10.5.1 Forward and Inverse Mapping

Geometric transforms define the relationship between the points in two images. This relation can be expressed in two ways. Either the coordinates of the output image, $\boldsymbol{x}'$, can be specified as a function of the input coordinates, $\boldsymbol{x}$, or vice versa:

$$\boldsymbol{x}' = M(\boldsymbol{x}) \quad \text{or} \quad \boldsymbol{x} = M^{-1}(\boldsymbol{x}'), \tag{10.31}$$

where M specifies the mapping function and M^{-1} its inverse. The two expressions in Eq. (10.31) give rise to two principal kinds of spatial transformation: *forward mapping* and *inverse mapping*.

With *forward mapping*, a pixel of the input image is mapped onto the output image (Fig. 10.15a). Generally, a pixel of the input image lies in-between the pixels of the output image. With forward mapping, it is not appropriate just to assign the value of the input pixel to the nearest pixel in the output image (point-to-point or nearest neighbor mapping). Then, it may happen that the transformed image contains holes as a value is never assigned to a pixel in the output image or that a value is assigned more than once to a point in the output image. An appropriate technique distributes the value of the input pixel to several output pixels. The easiest procedure is to regard pixels as squares and to take the fraction of the area of the input pixel that covers the output pixel as the weighting factor. Each output pixel accumulates the corresponding fractions of the input pixels which — if the mapping is continuous — add up to cover the whole output pixel.

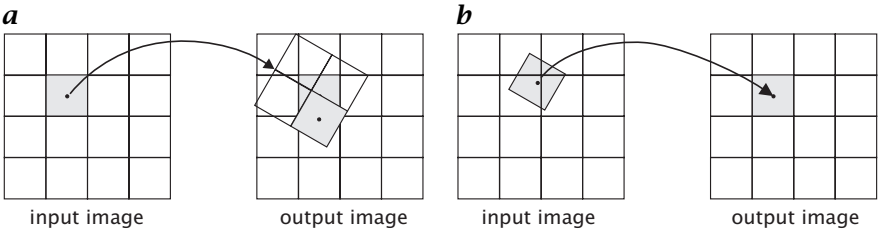

Figure 10.15: *Illustration of* ***a*** *forward mapping and* ***b*** *inverse mapping for spatial transformation of images.*

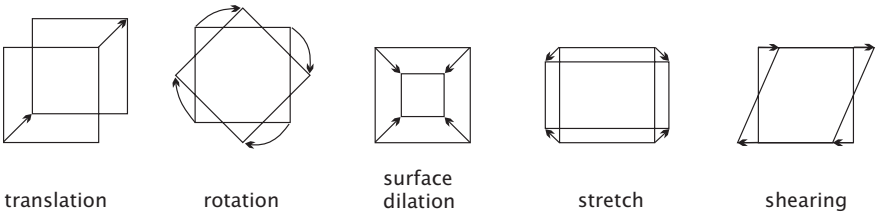

Figure 10.16: *Elementary geometric transforms for a planar surface element: translation, rotation, dilation, stretching, and shearing.*

With *inverse mapping*, the coordinates of a point in the output image are mapped back onto the input image (Fig. 10.15b). It is obvious that this scheme avoids holes and overlaps in the output image as all pixels are scanned sequentially. Now, the interpolation problem occurs in the input image. The coordinates of the output image in general do not hit a pixel in the input image but lie in between the pixels. Thus, its correct value must be interpolated from the surrounding pixels. Generally, inverse mapping is a more flexible technique as it is easier to implement various types of interpolation techniques.

10.5.2 Affine Transform

An affine transform is a linear coordinate transformation that includes the elementary transformations *translation*, *rotation*, *scaling*, *stretching*, and *shearing* (Fig. 10.16) and can be expressed by vector addition and matrix multiplication.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}. \tag{10.32}$$

With homogeneous coordinates (Section 7.3.3), the affine transform is written with a single matrix multiplication as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{10.33}$$

An affine transform has six degrees of freedom: two for translation (t_x, t_y) and one each for rotation, scaling, stretching, and shearing (a_{11}, a_{12}, a_{21}, and a_{22}). The affine transform maps a triangle into a triangle and a rectangle into a parallelogram. Therefore, it is also referred to as *three-point mapping*. Thus, it is obvious that the use of the affine transform is restricted. More general distortions such as the mapping of a rectangle into an arbitrary quadrilateral are not affine transforms.

10.5.3 Perspective Transform

Perspective projection is the basis of optical imaging as discussed in Section 7.3. The affine transform corresponds to parallel projection and can only be used as a model for optical imaging in the limit of a small field of view. The general perspective transform is most conveniently written with *homogeneous coordinates* (Section 7.3.3) as

$$\begin{bmatrix} w'x' \\ w'y' \\ w' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{bmatrix} \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} \quad \text{or} \quad X' = PX. \tag{10.34}$$

The two additional coefficients, a_{31} and a_{32}, in comparison to the affine transform Eq. (10.33), describe the perspective projection (compare Eq. (7.15) in Section 7.3.3).

Written in standard coordinates, the perspective transform according to Eq. (10.34) reads as

$$\begin{aligned} x' &= \frac{a_{11}x + a_{12}y + a_{13}}{a_{31}x + a_{32}y + 1} \\ y' &= \frac{a_{21}x + a_{22}y + a_{23}}{a_{31}x + a_{32}y + 1}. \end{aligned} \tag{10.35}$$

In contrast to the affine transform, the perspective transform is nonlinear. However, it is reduced to a linear transform by using homogeneous coordinates. A perspective transform maps lines into lines but only lines parallel to the projection plane remain parallel. A rectangle is mapped into an arbitrary quadrilateral. Therefore, the perspective transform is also referred to as *four-point mapping*.

10.5.4 Determination of Transform Coefficients by Point Matching

Generally, the coefficients of a transform, as described in Sections 10.5.2 and 10.5.3, are not known. Instead we have a set of corresponding points between the object and image space. In this section, we learn how to infer the coefficients of a transform from sets of corresponding points. For an affine transform, we need three non-collinear points (to map a triangle into a triangle). With these three points, Eq. (10.33) results in the following linear equation system:

$$\begin{bmatrix} x'_1 & x'_2 & x'_3 \\ y'_1 & y'_2 & y'_3 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \tag{10.36}$$

or

$$\boldsymbol{P}' = \boldsymbol{A}\boldsymbol{P} \tag{10.37}$$

from which $\boldsymbol{A}$ can be computed as

$$\boldsymbol{A} = \boldsymbol{P}'\boldsymbol{P}^{-1}. \tag{10.38}$$

The inverse of the matrix $\boldsymbol{P}$ exists when the three points $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, $\boldsymbol{X}_3$ are linearly independent. This means geometrically that they must not lie on one line.

With more than three corresponding points, the parameters of the affine transform can be solved by the following equation system in a least square sense (Section 17.6):

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{P}'\boldsymbol{P}^T(\boldsymbol{P}\boldsymbol{P}^T)^{-1} \quad \text{with} \\ \boldsymbol{P}'\boldsymbol{P}^T &= \begin{bmatrix} \sum x'_n x_n & \sum x'_n y_n & \sum x'_n \\ \sum y'_n x_n & \sum y'_n y_n & \sum y'_n \\ \sum x_n & \sum y_n & N \end{bmatrix} \\ \boldsymbol{P}\boldsymbol{P}^T &= \begin{bmatrix} \sum x_n^2 & \sum x_n y_n & \sum x_n \\ \sum x_n y_n & \sum y_n^2 & \sum y_n \\ \sum x_n & \sum y_n & N \end{bmatrix}. \end{aligned} \tag{10.39}$$

The inverse of an affine transform is itself affine. The transformation matrix of the inverse transform is given by the inverse of A^{-1}.

The determination of the coefficients for the perspective projection is slightly more complex. Given four or more corresponding points, the coefficients of the perspective transform can be determined. To that end, we rewrite Eq. (10.35) as

$$\begin{aligned} x' &= a_{11}x + a_{12}y + a_{13} - a_{31}xx' - a_{32}yx' \\ y' &= a_{21}x + a_{22}y + a_{23} - a_{31}xy' - a_{32}yy'. \end{aligned} \tag{10.40}$$

For N points, this leads to a linear equation system with $2N$ equations and 8 unknowns of the form

$$\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_N \\ y'_N \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 \\ & & & & \vdots & & & \\ x_N & x_N & 1 & 0 & 0 & 0 & -x_Nx'_N & -y_Nx'_N \\ 0 & 0 & 0 & x_N & y_N & 1 & -x_Ny'_N & -y_Ny'_N \end{bmatrix} \begin{bmatrix} a'_{11} \\ a'_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \end{bmatrix}$$

or

$$\boldsymbol{d} = \boldsymbol{M}\boldsymbol{a}. \tag{10.41}$$

This can be solved as a least square problem by

$$\boldsymbol{a} = (\boldsymbol{M}^T\boldsymbol{M})^{-1}\boldsymbol{M}^T\boldsymbol{d}. \tag{10.42}$$

10.6 Interpolation†

10.6.1 General

The other important aspect of discrete geometric operations besides the transform is *interpolation*. Interpolation is required as the transformed grid points of the input image in general no longer coincide with the grid points of the output image and vice versa.

The basis of interpolation is the sampling theorem (Section 9.2.2). This theorem states that the digital image *completely* represents the continuous image provided the sampling conditions are met. In short it means that each periodic structure that occurs in the image must be sampled at least twice per wavelength. From this basic fact it is easy — at least in principle — to devise a general framework for interpolation: reconstruct the continuous image first and then perform a new sampling to the new grid points. This procedure only works as long as the new grid has an equal or a narrower grid spacing. If it is wider, aliasing will occur. In this case, the image must be pre-filtered before it is resampled.

Although these procedures sound simple and straightforward, they are not at all. The problem is related to the fact that the reconstruction of the continuous image from the sampled image in practice is quite involved and can be performed only approximately. Thus, we need to consider how to optimize the interpolation given certain constraints. In this section, we will first see why ideal interpolation is not possible and then discuss various practical approaches in Sections 10.6.2–10.6.6.

In Section 9.3.1, we stated that reconstruction of a continuous function from sampled points can be considered as a convolution operation,

$$g_r(\boldsymbol{x}) = \sum_{m,n} g(\boldsymbol{x}_{m,n})h(\boldsymbol{x} - \boldsymbol{x}_{m,n}), \tag{10.43}$$

where the continuous interpolation mask h is the sinc function

$$h(\boldsymbol{x}) = \frac{\sin \pi x_1/\Delta x_1}{\pi x_1/\Delta x_1} \frac{\sin \pi x_2/\Delta x_2}{\pi x_2/\Delta x_2}. \tag{10.44}$$

The transfer function of the point spread function in Eq. (10.44) is a box function with width $\Delta x_w/2\pi$:

$$\hat{h}(\boldsymbol{k}) = \Pi(\tilde{k}_1/2, \tilde{k}_2/2) \quad \text{with} \quad \tilde{k}_w = k_w \Delta x_w/\pi. \tag{10.45}$$

The interpolatory nature of the convolution kernel Eq. (10.44) can be inferred from the following properties. The interpolated values in Eq. (10.43) at grid points $\boldsymbol{x}_{mn}$ should reproduce the grid points and not depend on any other grid point. From this condition, we can infer the *interpolation condition*:

$$h(\boldsymbol{x}_{m,n}) = \begin{cases} 1 & m = 0, n = 0 \\ 0 & \text{otherwise} \end{cases}. \tag{10.46}$$

The interpolation mask in Eq. (10.44) meets this interpolation condition. Any interpolation mask must, therefore, have zero crossings at all grid points except the zero point where it is 1.

The fact that interpolation is a convolution operation and thus can be described by a transfer function in Fourier space Eq. (10.45) gives us a handy tool to rate the errors associated with an interpolation technique. The box-type transfer function for the ideal interpolation function simply means that all wave numbers within the range of possible wave numbers $|k_w| \leq \Delta x_w/\pi$ experience neither a phase shift nor amplitude damping. Also, no wave numbers beyond the allowed interval are present in the interpolated signal, because the transfer function is zero there.

The ideal interpolation function in Eq. (10.43) is separable. Therefore, interpolation can as easily be formulated for higher-dimensional images. We can expect that all solutions to the interpolation problem will also be separable. Consequently, we need only discuss the 1-D interpolation problem. Once it is solved, we also have a solution for the n-dimensional interpolation problem.

An important special case is the interpolation to intermediate grid points halfway between the existing grid points. This scheme doubles the resolution and image size in all directions in which it is applied. Then, the continuous interpolation kernel reduces to a discrete convolution mask. As the interpolation kernel Eq. (10.44) is separable, we can first interpolate the intermediate points in a row in the horizontal direction before we apply vertical interpolation to the intermediate rows. In three dimensions, a third 1-D interpolation is added in the z or t direction. The interpolation kernels are the same in all directions. We need

the continuous kernel $h(x)$ only at half-integer values for $x/\Delta x$. From Eq. (10.44) we obtain the discrete ideal interpolation kernel

$$h = \left[\cdots \frac{(-1)^{m-1}\,2}{(2m-1)\pi} \cdots -\frac{2}{3\pi} \; \frac{2}{\pi} \; \frac{2}{\pi} \; -\frac{2}{3\pi} \cdots \frac{(-1)^{m-1}\,2}{(2m-1)\pi} \cdots \right] \tag{10.47}$$

with coefficients of alternating sign.

10.6.2 Interpolation in Fourier Space

Interpolation reduces to a simple operation in the Fourier domain. As shown by Eq. (10.45), the transfer function of an ideal interpolation kernel is a rectangular (box) function which is zero outside the wave numbers that can be represented. This basic fact suggests the following interpolation procedure in Fourier space:

1. Enlarge the matrix of the Fourier transformed image. If an $M \times M$ matrix is increased to an $M' \times M'$ matrix, the image in the spatial domain is also increased to an $M' \times M'$ image. Because of the reciprocity of the Fourier transform, the image size remains unchanged. Only the spacing between pixels is decreased, resulting in a higher spatial resolution:

$$M\Delta k \to M'\Delta k \quad \circ\!\!-\!\!\bullet \quad \Delta x = \frac{1}{M\Delta k} \to \Delta x' = \frac{1}{M'\Delta k} \tag{10.48}$$

2. Fill the padded area in the Fourier space with zeroes and compute an inverse Fourier transform.

Theoretically, this procedure results in a perfectly interpolated image. Unfortunately, it has three serious drawbacks.

1. The Fourier transform of a finite image implies a cyclic repetition of the image both in the spatial and Fourier domain. Thus, the convolution performed by the Fourier transform is also cyclic. This means that at the right or left edge of the image, convolution continues with the image at the opposite side. As the real world is not periodic and interpolation masks are large, this may lead to significant distortions of the interpolation even at quite large distances from the edges of the image.
2. The Fourier transform can be computed efficiently only for a specified number of values for M'. Best known are the fast radix-2 algorithms that can be applied only to images of the size $M' = 2^{N'}$ (Section 2.5.2). Therefore, the Fourier transform-based interpolation is limited to scaling factors of powers of two.
3. As the Fourier transform is a global transform, it can be applied only to scaling. In principle, it could also be applied to rotation and affine

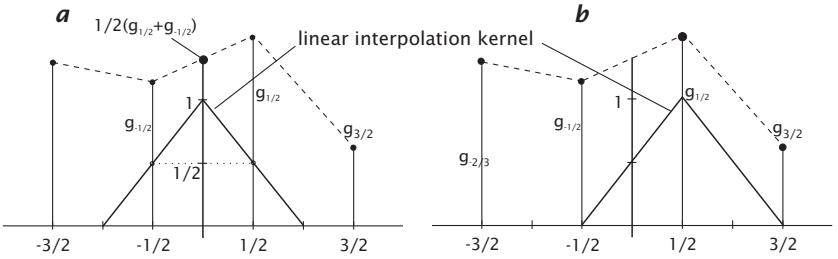

Figure 10.17: *Illustration of linear interpolation:* ***a*** *at $x = 0$, the mean of $g_{1/2}$ and $g_{-1/2}$ is taken,* ***b*** *at $x = 1/2$, $g_{1/2}$ is replicated.*

transforms. But then the interpolation problem is only shifted from the spatial domain to the wave number domain.

10.6.3 Linear Interpolation

Linear interpolation is the classic approach to interpolation. The interpolated points lie on pieces of straight lines connecting neighboring grid points. In order to simplify the expression, we use in the following normalized spatial coordinates $\tilde{x} = x/\Delta x$. We locate the two grid points at $-1/2$ and $1/2$. This yields the interpolation equation

$$g(\tilde{x}) = \frac{g_{1/2} + g_{-1/2}}{2} + (g_{1/2} - g_{-1/2})\,\tilde{x} \quad \text{for} \quad |\tilde{x}| \leq 1/2. \tag{10.49}$$

By comparison of Eq. (10.49) with Eq. (10.43), we can conclude that the continuous interpolation mask for linear interpolation is

$$h_1(\tilde{x}) = \begin{cases} 1 - |\tilde{x}| & |\tilde{x}| \leq 1 \\ 0 & \text{otherwise} \end{cases}. \tag{10.50}$$

Its interpolatory nature is illustrated in Fig. 10.17. The transfer function of the interpolation mask for linear interpolation, the triangle function $h_1(x)$ Eq. (10.50), is the squared sinc function (➤ R5)

$$\hat{h}_1(\tilde{k}) = \frac{\sin^2 \pi\tilde{k}/2}{(\pi\tilde{k}/2)^2}. \tag{10.51}$$

A comparison with the ideal transfer function for interpolation Eq. (10.45) shows that two distortions are introduced by linear interpolation:

1. While low wave numbers (and especially the mean value $\tilde{k} = 0$) are interpolated correctly, high wave numbers are slightly reduced in amplitude, resulting in some degree of smoothing. At $\tilde{k} = 1$, the transfer function is reduced to about 40%: $\hat{h}_1(1) = (2/\pi)^2 \approx 0.4$.

2. Since $\hat{h}_1(\tilde{k})$ is not zero at wave numbers $\tilde{k} > 1$, some spurious high wave numbers are introduced. If the continuously interpolated image is resampled, this yields moderate aliasing. The first sidelobe has an amplitude of $(2/3\pi)^2 \approx 0.045$.

If we interpolate only the intermediate grid points at $\tilde{x} = 0$, the continuous interpolation function Eq. (10.50) reduces to a discrete convolution mask with values at $\tilde{x} = [\ldots\ -3/2\ \ -1/2\ 1/2\ 3/2\ \ldots]$. As Eq. (10.50) is zero for $|\tilde{x}| \geq 1$, we obtain the simple interpolation mask $H = 1/2[11]$ with the transfer function

$$\hat{H}_1(\tilde{k}) = \cos \pi\tilde{k}/2. \tag{10.52}$$

The transfer function is real, so no phase shifts occur. The significant amplitude damping at higher wave numbers, however, shows that structures with high wave numbers are not correctly interpolated. Phase shifts do occur at all other values except for the intermediate grid points at $\tilde{x} = 0$. We investigate the phase shift and amplitude attenuation of linear interpolation at arbitrary fractional integer shifts $\epsilon \in [-1/2, 1/2]$. The interpolation mask for a point at ϵ is then $[1/2 - \epsilon, 1/2 + \epsilon]$. The mask contains a symmetric part $[1/2, 1/2]$ and an antisymmetric part $[-\epsilon, \epsilon]$. Therefore, the transfer function is complex and has the form

$$\hat{h}_1(\epsilon, \tilde{k}) = \cos \pi\tilde{k}/2 + 2\mathrm{i}\epsilon \sin \pi\tilde{k}/2. \tag{10.53}$$

In order to estimate the error in the phase shift, it is useful to compensate for the linear phase shift $\Delta\varphi = \epsilon\pi\tilde{k}$ caused by the displacement ϵ. According to the shift theorem (Theorem 3, p. 52, ➤ R4), it is required to multiply Eq. (10.53) by $\exp(-\mathrm{i}\epsilon\pi\tilde{k})$: Then we obtain

$$\hat{h}_1(\epsilon, \tilde{k}) = (\cos \pi\tilde{k}/2 + 2\mathrm{i}\epsilon \sin \pi\tilde{k}/2) \exp(-\mathrm{i}\epsilon\pi\tilde{k}). \tag{10.54}$$

Only for $\epsilon = 0$ and $\epsilon = 1/2$ is the transfer function real: $\hat{h}_1(0, \tilde{k}) = \cos \pi\tilde{k}/2$, $h_1(1/2, \tilde{k}) = 1$; but at all other fractional shifts, a non-zero phase shift remains, as illustrated in Fig. 10.18. The phase shift $\Delta\varphi$ is expressed as the position shift Δx of the corresponding periodic structure, i. e., $\Delta x = \Delta\varphi\lambda/2\pi = \Delta\varphi/(\pi\tilde{k})$.

10.6.4 Polynomial Interpolation

Given the significant limitations of linear interpolation as discussed in Section 10.6.3, we ask whether higher-order interpolation schemes perform better. The basic principle of linear interpolation was that a straight line was drawn to pass through two neighboring points. In the same way, we can use a polynomial of degree P that must pass through $P+1$ points with $P+1$ unknown coefficients a_p:

$$g_r(\tilde{x}) = \sum_{p=0}^{P} a_p \tilde{x}^p. \tag{10.55}$$

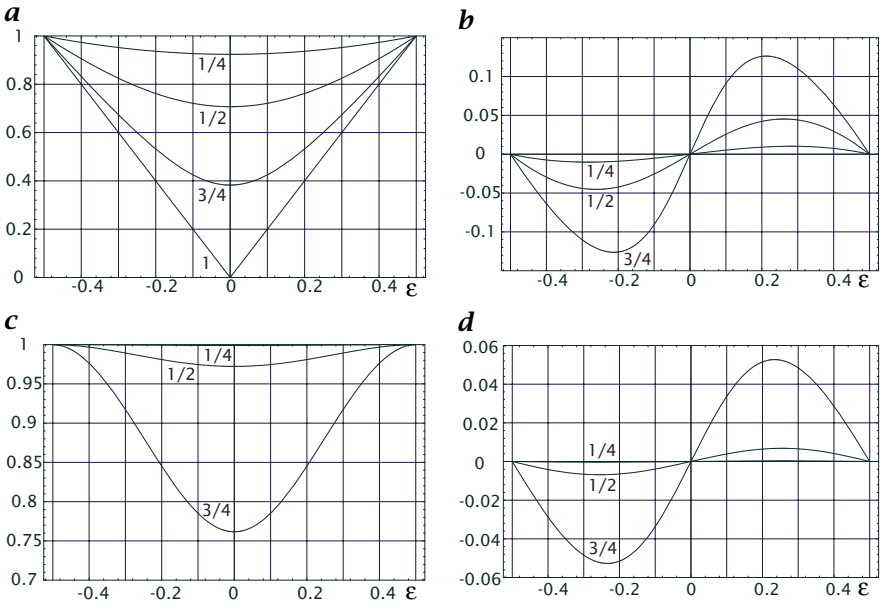

Figure 10.18: *Amplitude attenuation (left column) and phase shift expressed as a position shift* $\Delta x = \Delta\varphi\lambda/2\pi$ *(right column) in radians for wave numbers* $\tilde{k} = 1/4, 1/2, 3/4$*, as indicated, displayed as a function of the fractional position from* $-1/2$ *to* $1/2$ *for linear interpolation (**a** and **b**) and cubic B-spline interpolation (**c** and **d**).*

For symmetry reasons, in case of an even number of grid points, we set their position at half-integer values

$$\tilde{x}_p = \frac{2p - P}{2}. \tag{10.56}$$

From the interpolation condition at the grid points $g_r(\tilde{x}_p) = g_p$, we obtain a linear equation system with $P+1$ equations and $P+1$ unknowns a_P of the following form when P is odd:

$$\begin{bmatrix} g_0 \\ \vdots \\ g_{(P-1)/2} \\ g_{(P+1)/2} \\ \vdots \\ g_P \end{bmatrix} = \begin{bmatrix} 1 & -P/2 & P^2/4 & -P^3/8 & \cdots \\ \vdots & & & & \\ 1 & -1/2 & 1/4 & -1/8 & \cdots \\ 1 & 1/2 & 1/4 & 1/8 & \cdots \\ \vdots & & & & \\ 1 & P/2 & P^2/4 & P^3/8 & \cdots \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ \vdots \\ a_P \end{bmatrix} \tag{10.57}$$

from which we can determine the coefficients of the polynomial. For a cubic polynomial ($P = 3$), the equations system is

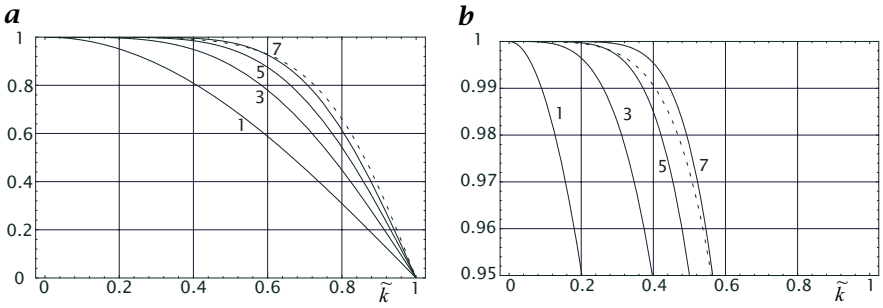

Figure 10.19: *Transfer function of discrete polynomial interpolation filters to interpolate the value between two grid points. The degree of the polynomial (1 = linear, 3 = cubic, etc.) is marked on the graph. The dashed line marks the transfer function for cubic B-spline interpolation (Section 10.6.5).* ***a*** *The full range,* ***b*** *a 5% margin below the ideal response* $\hat{h}(\tilde{k}) = 1$.

$$\begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 1 & -3/2 & 9/4 & -27/8 \\ 1 & -1/2 & 1/4 & -1/8 \\ 1 & 1/2 & 1/4 & 1/8 \\ 1 & 3/2 & 9/4 & 27/8 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{10.58}$$

with the solution

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \frac{1}{48} \begin{bmatrix} -3 & 27 & 27 & -3 \\ 2 & -54 & 54 & -2 \\ 12 & -12 & -12 & 12 \\ -8 & 24 & -24 & 8 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix}. \tag{10.59}$$

From this solution, we can infer, for example, that the point at $\tilde{x} = 0$ is interpolated by $g_r(0) = a_0 = -1/16g_0 + 9/16g_1 + 9/16g_2 - 1/16g_3$ corresponding to the interpolation mask $1/16[-1, 9, 9, -1]$.

Figure 10.19 shows the transfer functions for a polynomial interpolation of various degrees. With increasing degree P of the interpolating polynomial, the transfer function approaches the box function better. However, convergence is slow. For an accurate interpolation, we must take a large interpolation mask.

10.6.5 Spline-Based Interpolation‡

Besides its limited accuracy, polynomial interpolation has another significant disadvantage. The interpolated curve is not continuous at the grid points already in its first derivative. This is due to the fact that for each interval between grid points another polynomial is taken. Thus, only the interpolated function is continuous at the grid points but not the derivatives.

Splines avoid this disadvantage by additional constraints for the continuity of derivatives at the grid points. From the wide classes of splines, we will here discuss only one class, the *B-splines*. As B-splines are separable, it is sufficient to

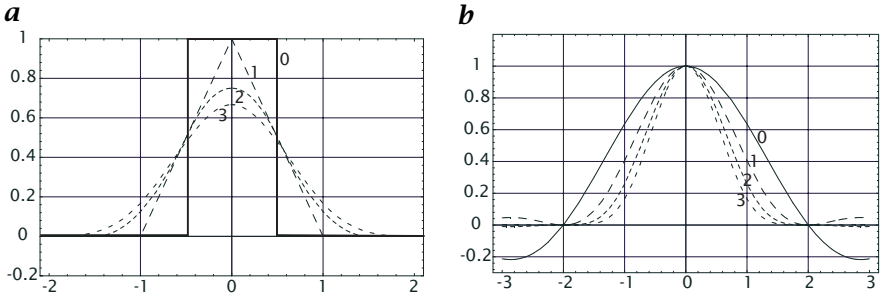

Figure 10.20: ***a*** *B-spline interpolation kernels generated by cascaded convolution of the box kernel of order 0 (nearest neighbor), 1 (linear interpolation), 2 (quadratic B-spline), and 3 (cubic B-spline);* ***b*** *corresponding transfer functions.*

discuss the properties of 1-D B-splines. From the background of image processing, the easiest access to B-splines is their convolution property. The kernel of a P-order B-spline curve is generated by convolving the box function $P+1$ times with itself (Fig. 10.20a):

$$\beta_P(\tilde{x}) = \underbrace{\Pi(\tilde{x}) * \ldots * \Pi(\tilde{x})}_{(P+1)\text{ times}}. \tag{10.60}$$

The transfer function of the box function is the sinc function (≻ R5). Therefore, the transfer function of the P-order B-spline is

$$\hat{\beta}_P(\hat{k}) = \left(\frac{\sin \pi\tilde{k}/2}{(\pi\tilde{k}/2)}\right)^{P+1}. \tag{10.61}$$

Figure 10.20b shows that the B-spline function does not make a suitable interpolation function. The transfer function decreases too early, indicating that B-spline interpolation performs too much averaging. Moreover, the B-spline kernel does not meet the interpolation condition Eq. (10.46) for $P > 1$.

B-splines can only be used for interpolation if first the discrete grid points are transformed in such a way that a following convolution with the B-spline kernel restores the original values at the grid points. This transformation is known as the B-spline transformation and constructed from the following condition:

$$g_p(x) = \sum_n c_n \beta_P(x - x_n) \quad \text{with} \quad g_p(x_n) = g(x_n). \tag{10.62}$$

If centered around a grid point, the B-spline interpolation kernel is unequal to zero only for three grid points. The coefficients $\beta_3(-1) = \beta_{-1}, \beta_3(0) = \beta_0$, and $\beta_3(1) = \beta_1$ are 1/6, 2/3, and 1/6. The convolution of this kernel with the unknown B-spline transform values c_n should result in the original values g_n at the grid points. Therefore,

$$\boldsymbol{g} = \boldsymbol{c} * \boldsymbol{\beta_3} \quad \text{or} \quad g_n = \sum_{n'=-1}^{1} c_{n+n'} \beta_{n'}. \tag{10.63}$$

Equation (10.63) constitutes the sparse linear equation system

$$\begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{N-1} \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 4 & 1 & 0 & \ddots & 0 & 1 \\ 1 & 4 & 1 & 0 & \ddots & 0 \\ 0 & 1 & 4 & 1 & 0 & \ddots \\ \ddots & & & & \ddots & \ddots \\ \ddots & \ddots & 1 & 4 & 1 & 0 \\ 0 & \ddots & 0 & 1 & 4 & 1 \\ 1 & 0 & \ddots & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{bmatrix} \tag{10.64}$$

using cyclic boundary conditions. The determination of the B-spline transformation thus requires the solution of a linear equation system with N unknowns. The special form of the equation system as a convolution operation, however, allows for a more efficient solution. In Fourier space, Eq. (10.63) reduces to

$$\hat{\boldsymbol{g}} = \hat{\boldsymbol{\beta}}_3 \hat{\boldsymbol{c}}. \tag{10.65}$$

The transfer function of $\boldsymbol{\beta}_3$ is $\hat{\beta}_3(\tilde{k}) = 2/3 + 1/3\cos(\pi\tilde{k})$. As this function has no zeroes, we can compute $\boldsymbol{c}$ by inverse filtering (Section 4.2.10), i. e., convoluting $\boldsymbol{g}$ with a mask that has the transfer function

$$\hat{\beta}_3^{-1}(\tilde{k}) = \hat{\beta}_T(\tilde{k}) = \frac{1}{2/3 + 1/3\cos\pi\tilde{k}}. \tag{10.66}$$

Such a transfer function is a kind of recursive filter (Section 4.2.10) that is applied first in the forward and then in the backward direction with the following recursion [187]:

$$\begin{aligned} g'_n &= g_n - (2-\sqrt{3})(g'_{n-1} - g_n) \\ c'_n &= g'_n - (2-\sqrt{3})(c_{n+1} - g'_n). \end{aligned} \tag{10.67}$$

The whole operation takes only two multiplications and four additions.

The B-spline interpolation is applied after the B-spline transformation. In the continuous case, using Eqs. (10.61) and (10.66), this yields the effective transfer function

$$\hat{\beta}_I(\tilde{k}) = \frac{\sin^4(\pi\tilde{k}/2)/(\pi\tilde{k}/2)^4}{(2/3 + 1/3\cos\pi\tilde{k})}. \tag{10.68}$$

Essentially, the B-spline transformation performs an amplification of high wave numbers (at $\tilde{k} = 1$ by a factor 3) which compensates the smoothing of the B-spline interpolation to a large extent.

We investigate this compensation at the grid points and at the intermediate points. From the equation of the cubic B-spline interpolating kernel Eq. (10.60) (see also Fig. 10.20a) the interpolation coefficients for the grid points and intermediate grid points are

$$\begin{aligned} &1/6\,[1\ 4\ 1] \quad \text{and} \\ &1/48\,[1\ 23\ 23\ 1], \end{aligned} \tag{10.69}$$

respectively. Therefore, the transfer functions are

$$\begin{aligned} &2/3 + 1/3\cos\pi\tilde{k} \quad \text{and} \\ &23/24\cos(\pi\tilde{k}/2) + 1/24\cos(3\pi\tilde{k}/2), \end{aligned} \tag{10.70}$$

respectively. At the grid points, the transfer function exactly compensates — as expected — the application of the B-spline transformation Eq. (10.66). Thus, the interpolation curve goes through the values at the grid points. At the intermediate points the effective transfer function for the cubic B-spline interpolation is then

$$\hat{\beta}_I(1/2,\tilde{k}) = \frac{23/24\cos(\pi\tilde{k}/2) + 1/24\cos(3\pi\tilde{k}/2)}{2/3 + 1/3\cos\pi\tilde{k}}. \tag{10.71}$$

The amplitude attenuation and the phase shifts expressed as a position shift in pixel distances are shown in Fig. 10.18c, d. Note that the shift is related to the intermediate grid. The shift and amplitude damping is zero at the grid points $[-0.5, 0.5]^T$. While the amplitude damping is maximal for the intermediate point, the position shift is also zero at the intermediate point because of symmetry reasons. Also, at the wave number $\tilde{k} = 3/4$ the phase shift is unfortunately only about two times smaller than for linear interpolation (Fig. 10.18b). It is still significant with a maximum of about 0.13.

This value is much too high for algorithms that ought to be accurate in the 1/100 pixel range. If no better interpolation technique can be applied, this means that the maximum wave number should be lower than 0.5. Then, the maximum shift is lower than 0.01 and the amplitude damping less than 3 %.

Note that these comments on phase shifts only apply for arbitrary fractional shifts. For pixels on the intermediate grid, no position shift occurs at all. In this special case — which often occurs in image processing, for instance for pyramid computations (Chapter 5) — optimization of interpolation filters is quite easy because only the amplitude damping must be minimized over the wave number range of interest.

10.6.6 Optimized Interpolation‡

Filter design for interpolation — like any filter design problem — can be treated in a mathematically more rigorous way as an optimization problem. The general idea is to vary the filter coefficients in such a way that the derivation from the ideal transfer function reaches a minimum. For non-recursive filters, the transfer function is linear in the cofficients h_r:

$$\hat{h}(\tilde{k}) = \sum_{r=1}^{R} h_r \hat{f}_r(\tilde{k}). \tag{10.72}$$

Let the ideal transfer function be $\hat{h}_I(\tilde{k})$. Then the optimization procedure should minimize the integral

$$\int_0^1 w(\tilde{k}) \left| \left(\sum_{r=1}^{R} h_r \hat{f}_r(\tilde{k}) \right) - \hat{h_I}(\tilde{k}) \right|^n \mathrm{d}\tilde{k}. \tag{10.73}$$

In this expression, a weighting function $w(\tilde{k})$ has been introduced which allows control over the optimization for a certain wave number range. In equation

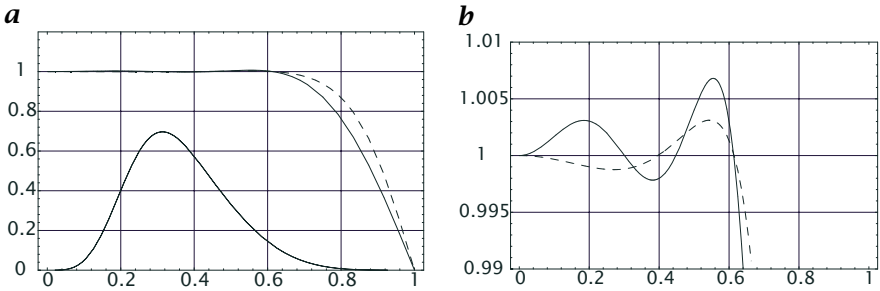

Figure 10.21: *Transfer function of interpolation kernels optimized with the weighted least squares technique of Eq. (10.76) and Eq. (10.77) with R = 3 (solid line) and of Eq. (10.78) for R = 2 (dashed line). The weighting function used for the optimization is shown in* ***a*** *as a thin solid line;* ***b*** *shows a narrow sector of the plot in* ***a*** *for a better estimation of small deviations from ideal values.*

Eq. (10.73) an arbitrary L_n-norm is included. Mostly the L_2-norm is taken, which minimizes the sum of squares.

For the L_2-norm, the minimization problem results in a linear equation system for the R coefficients of the filter which can readily be solved:

$$\boldsymbol{M}\boldsymbol{h} = \boldsymbol{d} \tag{10.74}$$

with

$$\boldsymbol{d} = \begin{bmatrix} \overline{h_I \hat{f}_1} \\ \overline{h_I \hat{f}_2} \\ \vdots \\ \overline{h_I \hat{f}_R} \end{bmatrix} \quad \text{and} \quad \boldsymbol{M} = \begin{bmatrix} \overline{\hat{f}_1^2} & \overline{\hat{f}_1 \hat{f}_2} & \cdots & \overline{\hat{f}_1 \hat{f}_R} \\ \overline{\hat{f}_1 \hat{f}_2} & \overline{\hat{f}_2^2} & \cdots & \overline{\hat{f}_2 \hat{f}_R} \\ \vdots & & \ddots & \vdots \\ \overline{\hat{f}_1 \hat{f}_R} & \overline{\hat{f}_2 \hat{f}_R} & \cdots & \overline{\hat{f}_R^2} \end{bmatrix}$$

where the abbreviation

$$\overline{\hat{e}(\tilde{k})} = \int_0^1 w(\tilde{k}) \cdot \hat{e}(\tilde{k}) \mathrm{d}\tilde{k} \tag{10.75}$$

for an arbitrary function $e(\tilde{k})$ has been used.

The flexibility of the least squares optimization technique for filter design is given by the free choice of the weighting function $w(\tilde{k})$ and the careful consideration of the symmetry properties and other features of a filter by the choice of the transfer function in Eq. (10.72). For illustration, we discuss the following two approaches:

$$\hat{h}(\tilde{k}) = \sum_{r=1}^{R} h_r \cos\left(\frac{2r-1}{2}\pi\tilde{k}\right) \tag{10.76}$$

and

$$\hat{h}(\tilde{k}) = \cos\left(\frac{1}{2}\pi\tilde{k}\right) + \sum_{r=2}^{R} h_r \left[\cos\left(\frac{2r-3}{2}\pi\tilde{k}\right) - \cos\left(\frac{1}{2}\pi\tilde{k}\right)\right]. \tag{10.77}$$

Both filters result in a symmetric mask by the choice of the cosine function. Equation Eq. (10.77) ensures that $\hat{h}(0) = 1$, i. e., the mean gray values are preserved by the interpolation. This is done by forcing the first coefficient, h_1, to

be one minus the sum of all others. Equation Eq. (10.76) does not apply this constraint. Figure 10.21 compares the optimal transfer functions with both approaches for $R = 3$.

The filters are significantly better than those obtained by polynomial and cubic B-spline interpolation (Fig. 10.19). The additional degree of freedom for Eq. (10.76) leads to significantly better solutions for the wave number range where the weighting function is maximal.

Even better interpolation masks can be obtained by using a combination of nonrecursive and recursive filters, as with the cubic B-spline interpolation:

$$\hat{h}(\tilde{k}) = \frac{\cos\left(1/2\,\pi\tilde{k}\right) + \sum_{r=2}^{R} h_r \left[\cos\left((2r-3)/2\,\pi\tilde{k}\right) - \cos\left(1/2\,\pi\tilde{k}\right)\right]}{1 - \alpha + \alpha\cos\left(\pi\tilde{k}\right)}. \quad (10.78)$$

With recursive filters, the least squares optimization becomes nonlinear because $\hat{h}(\tilde{k})$ in Eq. (10.78) is nonlinear in the parameter α of the recursive filter. Then, iterative techniques are required to solve the optimization problem. Figure 10.21c, d shows the transfer functions for $R = 2$. A more detailed discussion of interpolation filters including tables with optimized filters can be found in Jähne [81].

10.6.7 Fast Algorithms for Geometric Transforms‡

With the extensive discussion on interpolation we are well equipped to devise fast algorithms for the different geometric transforms. Basically, all fast interpolation algorithms use the following two principles: efficient computation employing the interpolation coefficients and partition into 1-D geometric transforms.

First, many computations are required to compute the interpolation coefficients for fractional shifts. For each shift, different interpolation coefficients are required. Thus we must devise the transforms in such a way that we need only constant shifts for a certain pass of the transform. If this is not possible, it might still be efficient to precompute the interpolation coefficients for different fractional shifts and to save them for later usage.

Second, we learnt in Section 10.6.1 that interpolation is a separable procedure. Taking advantage of this basic fact considerably reduces the number of operations. In most cases it is possible to separate the two- and higher dimensional geometric transforms into a series of 1-D transforms.

10.7 Further Readings‡

Holst [70, 72] and Biberman [7] deal with radiometric calibration of sensors and cameras in the visible and infrared. A detailed discussion of interpolation filters including tables with filter coefficients and efficient algorithms for geometric transforms can be found in Jähne [81, Chap. 8]. Readers interested in the mathematical background of interpolation are referred to Davis [25] and Lancaster and Salkauskas [103]. Wolberg [200] expounds geometric transforms.

Part III

Feature Extraction

11 Averaging

11.1 Introduction

In this chapter we will discuss neighborhood operations for performing the elementary task of averaging. This operation is of central importance for low-level image processing. It is one of the building blocks for the more complex feature extraction operators discussed in Chapters 13–15.

In the simplest case, objects are identified as *regions* of constant radiance, i. e., gray values. Then, averaging gives adequate mean values of the gray values within the object. This approach, of course, implies a simple model of the image content. The objects of interest must indeed be characterized by constant gray values that are clearly different from the background and/or other objects.

However, this assumption is seldom met in real-world applications. The intensities will generally show some variations. These variations may be an inherent feature of the object or could be caused by the image formation process. Typical cases are *noise*, a *non-uniform illumination*, or *inhomogeneous background*.

In complex cases, it is not possible to distinguish objects from the background with just one feature. Then it may be a valid approach to compute more than one feature image from one and the same image. This results in a multicomponent or *vectorial feature image*.

The same situation arises when more than one image is taken from a scene as with *color images* or any type of *multispectral image*. Therefore, the task of averaging must also be applied to vectorial images. In image sequences, averaging is extended into the time coordinate to a spatiotemporal averaging.

11.2 General Properties of Averaging Filters

Convolution provides the framework for all elementary averaging filters. These filters have a number of properties in common that are discussed in this section.

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2

Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

11.2.1 Zero Shift

With respect to object detection, the most important feature of a smoothing convolution operator is that it must not shift the object position. Any shift introduced by a preprocessing operator would cause errors in the estimates of the position and possibly other geometric features of an object. In order to cause no shift, the transfer function of a filter must be real. A filter with this property is known as a *zero-phase filter*, because it does not introduce a phase shift in any of the periodic components of an image. A real transfer function implies a symmetric filter mask (Section 2.3). A W-dimensional symmetric convolution mask is defined by

$$\begin{aligned} &\text{1-D:} \quad h_{-n} = h_n \\ &\text{2-D:} \quad h_{-m,n} = h_{m,n},\ h_{m,-n} = h_{m,n} \\ &\text{3-D:} \quad h_{-l,m,n} = h_{l,m,n},\ h_{l,-m,n} = h_{l,m,n},\ h_{l,m,-n} = h_{l,m,n}. \end{aligned} \tag{11.1}$$

The symmetry relations also significantly ease the computation of the transfer functions as only the cosine term of the complex exponential from the Fourier transform remains in the equations. The transfer function for 1-D symmetric masks with an odd number of coefficients $(2R+1)$ is

$$\hat{h}(\tilde{k}) = h_0 + 2\sum_{v=1}^{R} h_v \cos(v\pi\tilde{k}). \tag{11.2}$$

With an even number of coefficients $(2R)$, the transfer function of a 1-D symmetric mask is given by

$$\hat{h}(\tilde{k}) = 2\sum_{v=1}^{R} h_v \cos((v-1/2)\pi\tilde{k}). \tag{11.3}$$

Note that the wave numbers are half-integers $v = 1/2, 3/2, \ldots$, because for symmetry reasons the result of the convolution with an even-sized mask lies on the intermediate grid.

For a 2-D symmetric mask with an odd number of coefficients in both directions we obtain correspondingly:

$$\begin{aligned} \hat{h}(\tilde{\boldsymbol{k}}) &= h_{00} \\ &+ 2\sum_{v=1}^{r} h_{0v}\cos(v\pi\tilde{k}_1) + \sum_{u=1}^{R} h_{u0}\cos(u\pi\tilde{k}_2) \\ &+ 4\sum_{u=1}^{R}\sum_{v=1}^{R} h_{uv}\cos(v\pi\tilde{k}_1)\cos(u\pi\tilde{k}_2). \end{aligned} \tag{11.4}$$

A further discussion of the properties of symmetric masks up to three dimensions can be found in Jähne [81].

11.2.2 Preservation of Mean

The mean value must be preserved by a smoothing operator. This condition says that the transfer function for the zero wave number is 1 or, equivalently, that the sum of all coefficients of the mask is 1:

$$\begin{aligned} &\text{1-D:} \quad \hat{h}(0) = 1 \quad \sum_n h_n = 1 \\ &\text{2-D:} \quad \hat{h}(\mathbf{0}) = 1 \quad \sum_m \sum_n h_{mn} = 1 \\ &\text{3-D:} \quad \hat{h}(\mathbf{0}) = 1 \quad \sum_l \sum_m \sum_n h_{lmn} = 1. \end{aligned} \tag{11.5}$$

11.2.3 Monotonically Decreasing Transfer Function

Intuitively, we expect that any smoothing operator attenuates smaller scales more strongly than coarser scales. More specifically, a smoothing operator should not completely annul a certain scale while smaller scales still remain in the image. Mathematically speaking, this means that the transfer function decreases monotonically with the wave number:

$$\hat{h}(\tilde{k}_2) \leq \hat{h}(\tilde{k}_1) \quad \text{if} \quad \tilde{k}_2 > \tilde{k}_1. \tag{11.6}$$

We may impose the more stringent condition that for the highest wave numbers the transfer function is identical to zero:

$$\begin{aligned} &\text{1-D:} \quad \hat{h}(1) = 0 \\ &\text{2-D:} \quad \hat{h}(\tilde{k}_1, 1) = 0, \quad \hat{h}(1, \tilde{k}_2) = 0 \\ &\text{3-D:} \quad \hat{h}(\tilde{k}_1, \tilde{k}_2, 1) = 0, \quad \hat{h}(\tilde{k}_1, 1, \tilde{k}_3) = 0, \quad \hat{h}(1, \tilde{k}_2, \tilde{k}_3) = 0. \end{aligned} \tag{11.7}$$

Together with the monotonicity condition and the preservation of the mean value, this means that the transfer function decreases monotonically from one to zero for each averaging operator.

11.2.4 Isotropy

In most applications, the smoothing should be the same in all directions in order not to prefer any direction. Thus, both the filter mask and the transfer function should be isotropic. Consequently, the filter mask depends only on the magnitude of the distance from the center pixel and the transfer function on the magnitude of the wave number:

$$h(\boldsymbol{x}) = h(|\boldsymbol{x}|) \quad \text{and} \quad \hat{h}(\tilde{\boldsymbol{k}}) = \hat{h}(|\tilde{\boldsymbol{k}}|). \tag{11.8}$$

In discrete space, of course, this condition can only be met approximately. Therefore, it is an important design goal to construct discrete masks with minimum deviation from isotropy.

11.3 Box Filter

11.3.1 Basics

It is obvious that smoothing filters will average pixels within a small neighborhood. The simplest method is to add all the pixels within the filter mask and to divide the sum by the number of pixels. Such a simple filter is called a *box filter*. Box filters are an illustrative example as to how to design a filter properly. As an introduction, we consider a 1×3 box filter

$$ {}^3\boldsymbol{R} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}. \tag{11.9} $$

The factor $1/3$ scales the result of the convolution sum in order to preserve the mean value (Section 11.2.2). Otherwise the gray value in a region with constant gray values is not preserved. We apply this mask to a vertical edge

$$ \begin{matrix} \vdots & \vdots & \vdots & \vdots & \\ \cdots & 0 & 0 & 1 & 1 & \cdots \\ \cdots & 0 & 0 & 1 & 1 & \cdots \\ \cdots & 0 & 0 & 1 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{matrix} * \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{matrix} \vdots & \vdots & \vdots & \vdots & \\ \cdots & 0 & 1/3 & 2/3 & 1 & \cdots \\ \cdots & 0 & 1/3 & 2/3 & 1 & \cdots \\ \cdots & 0 & 1/3 & 2/3 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{matrix} $$

As expected for a smoothing operation, the sharp edge is transformed into a smoother ramp with a gradual transition from 0 to 1. Smoothing filters attenuate structures with high wave numbers. Let us test this first with a vertical structure with a wavelength of 3 pixel distance:

$$ \begin{matrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ 1 & -2 & 1 & 1 & -2 & 1 & \cdots \\ 1 & -2 & 1 & 1 & -2 & 1 & \cdots \\ 1 & -2 & 1 & 1 & -2 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \end{matrix} * \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{matrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \end{matrix} $$

It turns out that the 1×3 box filter completely removes a structure with the wavelength 3. As already discussed in Section 11.2.3, we expect that all structures with a wave number above a certain threshold are removed by a good smoothing filter. This is not the case for the 1×3 box filter. A structure with the wavelength 2 is only attenuated by a factor of 3:

$$ \begin{matrix} \vdots & \vdots & \vdots & \vdots & \\ \cdots & 1 & -1 & 1 & -1 & \cdots \\ \cdots & 1 & -1 & 1 & -1 & \cdots \\ \cdots & 1 & -1 & 1 & -1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{matrix} * \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{matrix} \vdots & \vdots & \vdots & \vdots & \vdots & \\ \cdots & -1/3 & 1/3 & -1/3 & 1/3 & \cdots \\ \cdots & -1/3 & 1/3 & -1/3 & 1/3 & \cdots \\ \cdots & -1/3 & 1/3 & -1/3 & 1/3 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{matrix} $$

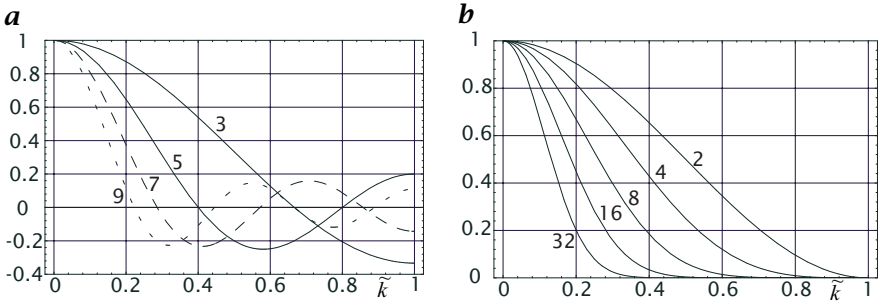

Figure 11.1: *Transfer functions of one-dimensional smoothing filters:* ***a*** *box filters with 3, 5, 7, and 9 coefficients;* ***b*** *binomial filters $\mathcal{B}^p$ with $p = 2$, 4, 8, 16, and 32.*

11.3.2 1-D Box Filter

After this qualitative introduction, we discuss box filters quantitatively by computing the transfer function. For sake of simplicity, we start with 1-D filters. The mask of the box filter

$$^3\boldsymbol{R}_x = \begin{bmatrix} 1/3 & 1/3 & 1/3 \end{bmatrix} \tag{11.10}$$

is of even symmetry. According to the considerations in Section 4.2.6, we can apply Eq. (4.25) to compute the transfer function of the one-dimensional 1×3 box filter. Only the coefficients $h_0 = h_1 = 1/3$ are unequal to zero and the transfer function reduces to

$$^3\hat{r}_x = \frac{1}{3} + \frac{2}{3}\cos(\pi\tilde{k}_x). \tag{11.11}$$

The transfer function is shown in Fig. 11.1a. Our quick computation at the beginning of this section is verified. The transfer function shows a zero at $\tilde{k} = 2/3$. This corresponds to a wave number which is sampled 3 times per wavelength. The smallest possible wavelength ($\tilde{k} = 1$), which is sampled twice per wavelength, is only damped by a factor of three. The transfer function is negative for $\tilde{k} > 2/3$. A negative transfer function means an interchange of minima and maxima, equal to a phase shift of 180°. In conclusion, the 1×3 box filter is not a good lowpass filter. It is disturbing that the attenuation does not increase monotonically with the wave number but oscillates. Even worse, structures with the largest wave number are not attenuated strongly enough.

Larger box filters do not show a significant improvement (Fig. 11.1a). On the contrary, the oscillatory behavior is more pronounced and the attenuation is only proportional to the wave number. For large filter masks, the discrete mask with R coefficients comes close to a continuous box function of width R. Therefore the transfer function approximates

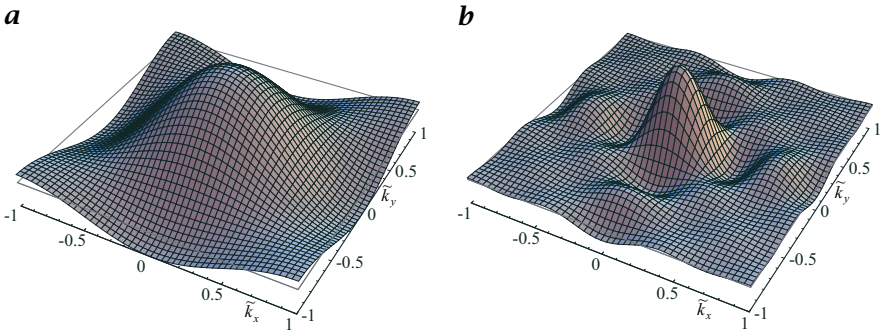

Figure 11.2: *Transfer functions of two-dimensional box filters shown in a pseudo 3-D plot:* ***a*** 3×3 *box filter;* ***b*** 7×7 *box filter.*

the sinc function (≻ R5):

$$ {}^{R}\hat{r}_x(\tilde{k}) = \frac{\sin(\pi R\tilde{k}/2)}{R\sin(\pi\tilde{k}/2)} \approx \frac{\sin(\pi R\tilde{k}/2)}{\pi R\tilde{k}/2} = \operatorname{sinc}(R\tilde{k}/2). \tag{11.12} $$

11.3.3 2-D Box Filter

Now we turn to two-dimensional box filters. To simplify the arithmetic, we utilize the fact that the filter is separable and decompose it into vertical and horizontal 1-D components, respectively:

$$ {}^{3}\boldsymbol{R} = {}^{3}\boldsymbol{R}_x * {}^{3}\boldsymbol{R}_y = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} * \frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. $$

The transfer function of the one-dimensional filters is given by Eq. (11.11) (replacing $\tilde{k}_x$ by $\tilde{k}_y$ for the vertical filter). As convolution in the space domain corresponds to multiplication in the wave number domain, the transfer function of $\boldsymbol{R}$ is

$$ {}^{3}\hat{r} = {}^{3}\hat{r}_x\,{}^{3}\hat{r}_y = \left[\frac{1}{3} + \frac{2}{3}\cos(\pi\tilde{k}_x)\right]\left[\frac{1}{3} + \frac{2}{3}\cos(\pi\tilde{k}_y)\right]. \tag{11.13} $$

11.3.4 Evaluation

From Eq. (11.13) and Fig. 11.2a, we can conclude that 2-D box filters are also poor lowpass filters. A larger box filter, for example one with a 7×7 mask (Fig. 11.2b), does not perform any better. Besides the disadvantages already discussed for the one-dimensional case, we are faced with the problem that the transfer function is not *isotropic*, i. e., it depends, for a given wave number, on the direction of the wave number.

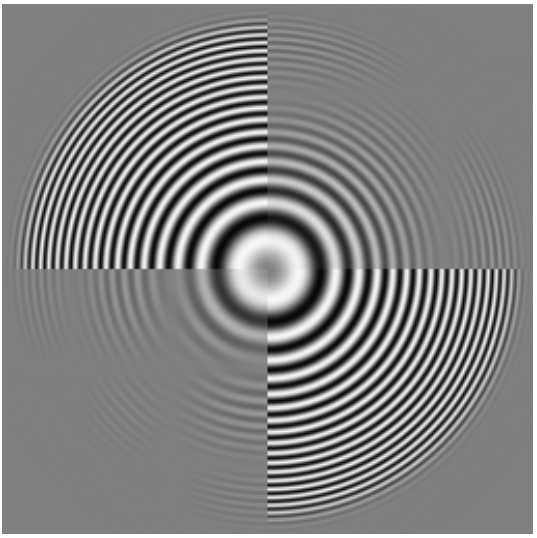

Figure 11.3: *Test of the smoothing with a* 5×5 *(upper right quadrant) and a* 9×9 *box filter (lower left quadrant) using a test image with concentric sinusoidal rings. The maximum wave number* $\tilde{k}$ *at the edge of the pattern is 0.6.*

When we apply a box filter to an arbitrary image, all these disadvantages affect the image, but it is difficult to observe them quantitatively (Fig. 11.6). They are revealed immediately, however, if we use a carefully designed *test image*. This image contains concentric sinusoidal rings. The wavelength of the rings decreases with distance from the center. With this test image, we map the Fourier domain onto the space domain. Thus, we can directly see the transfer function, i. e., the change in the amplitude and the phase shift, when a filter is applied. When we convolve this image with a 5×5 or 9×9 box filter, the deviations from an isotropic transfer function become readily visible (Fig. 11.3). We can observe the wave numbers that vanish entirely and the change of gray value maxima into gray value minima and vice versa in some regions, indicating the 180° phase shift caused by negative values in the transfer function.

From this experience, we can learn an important lesson. We must not rate the properties of a filter operation from its effect on arbitrary images, even if we think that they seem to work correctly. Obviously, the eye perceives a rather qualitative impression, but for quantitative extraction of image features a quantitative analysis of the filter properties is required. This involves a careful analysis of the transfer function and the application of the filters to carefully designed test images.

Now we turn back to the question of what went wrong with the box filter. We might try to design a better smoothing filter directly in the wave

number space. An ideal smoothing filter would cut off all wave numbers above a certain threshold value. We could use this ideal transfer function and compute the filter mask by an inverse Fourier transform. However, we run into two problems which can be understood without explicit calculations. The inverse Fourier transform of a box function is a sinc function. This means that the coefficients decrease only proportionally to the distance from the center pixel. We would be forced to work with large filter masks. Furthermore, the filter has the disadvantage that it overshoots at the edges.

11.3.5 Fast Computation

Despite all the disadvantages of box filters, they show one significant advantage. According to the following equation, the convolution with a one-dimensional box filter can be computed independently of its size with only three operations as a recursive filter operation:

$$g'_m = g'_{m-1} + \frac{1}{2r+1}(g_{m+r} - g_{m-r-1}). \tag{11.14}$$

This recursion can be understood by comparing the computations for the convolution at neighboring pixels. When the box mask is moved one position to the right, it contains the same weighting factor for all pixels except for the last and the first pixel. Thus, we can simply take the result of the previous convolution, (g'_{m-1}), subtract the first pixel that just moved out of the mask, (g_{m-r-1}), and add the gray value at the pixel that just came into the mask, (g_{m+r}). In this way, the computation of a box filter does not depend on its size and the number of computations is $O(r^0)$. Only one addition, one subtraction, and one multiplication are required to compute the filter result.

11.4 Binomial Filter

11.4.1 Basics

From our experience with box filters, we conclude that the design of filters is a difficult optimization problem. If we choose a small rectangular filter mask, we get a poor transfer function. If we start with an ideal transfer function, we get large filter masks and overshooting filter responses. The reason for this behavior is the fundamental relation between smoothness and compactness of the Fourier transform pairs (Section 2.3.5). An edge constitutes a discontinuity. A discontinuity leads to an impulse in the first derivative. The Fourier transform of an

impulse is evenly spread over the whole Fourier domain. Using the integral property of the Fourier transform (Section 2.3), an integration of the derivative in the space domain means a division by $\boldsymbol{k}$ in the Fourier domain (≻ R5). Then we know without any detailed calculation that in the one-dimensional case the envelope of the Fourier transform of a function which shows discontinuities in the space domain will decline with k^{-1} in the wave number domain. This was exactly what we found for the box function. Its Fourier transform is the sinc function (≻ R5).

Considering this basic fact, we can design better smoothing filters. One condition is that the filter masks should gradually approach zero.

11.4.2 1-D Binomial Filter

Here we will introduce a class of smoothing filters that meets this criterion and can be calculated very efficiently. Furthermore, these filters are an excellent example of how more complex filters can be built from simple components. The simplest and most elementary smoothing mask we can think of is

$$\boldsymbol{B} = \frac{1}{2}[1\ 1]. \tag{11.15}$$

It averages the gray values of two neighboring pixels. We can use this mask p times in a row on the same image. This corresponds to the filter mask

$$\frac{1}{2^p}\underbrace{[1\ 1] * [1\ 1] * \ldots * [1\ 1]}_{p \text{ times}}, \tag{11.16}$$

or, written as an operator equation,

$$\mathcal{B}^p = \underbrace{\mathcal{B}\mathcal{B}\ldots\mathcal{B}}_{p \text{ times}}. \tag{11.17}$$

Some examples of the resulting filter masks are:

$$\begin{aligned}
\boldsymbol{B}^2 &= 1/4\,[1\ 2\ 1] \\
\boldsymbol{B}^3 &= 1/8\,[1\ 3\ 3\ 1] \\
\boldsymbol{B}^4 &= 1/16\,[1\ 4\ 6\ 4\ 1] \\
\boldsymbol{B}^8 &= 1/256\,[1\ 8\ 28\ 56\ 70\ 56\ 28\ 8\ 1].
\end{aligned} \tag{11.18}$$

Because of symmetry, only the odd-sized filter masks are of interest. In order to perform a convolution with the asymmetric mask $1/2\,[1\ 1]$ correctly, we store the result in the right and left pixel alternately.

The masks contain the values of the discrete *binomial distribution*. Actually, the iterative composition of the mask by consecutive convolution with the $1/2\,[1\ 1]$ mask is equivalent to the computation scheme of

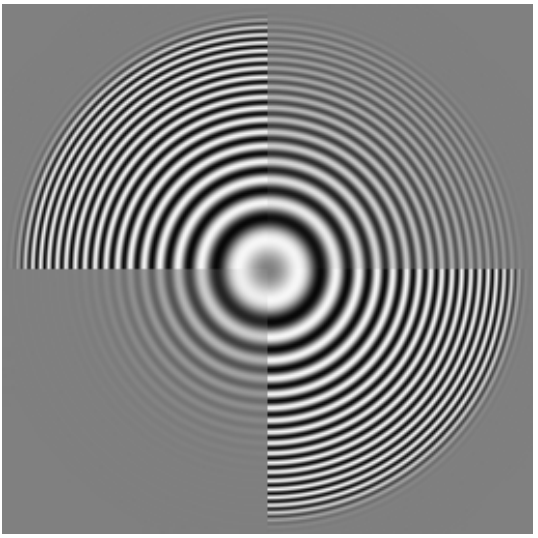

Figure 11.4: *Test of the smoothing with a $\mathcal{B}^4$ and $\mathcal{B}^{16}$ binomial filter using a test image with concentric sinusoidal rings.*

Pascal's triangle:

p	f		σ^2
0	1	1	0
1	1/2	1 1	1/4
2	1/4	1 2 1	1/2
3	1/8	1 3 3 1	3/4
4	1/16	1 4 6 4 1	1
5	1/32	1 5 10 10 5 1	5/4
6	1/64	1 6 15 20 15 6 1	3/2
7	1/128	1 7 21 35 35 21 7 1	7/4
8	1/256	1 8 28 56 70 56 28 8 1	2

(11.19)

where p denotes the order of the binomial, f the scaling factor 2^{-p}, and σ^2 the variance, i. e., effective width, of the mask.

The computation of the transfer function of a binomial mask is also very simple since we only need to know the transfer function of $\mathcal{B}$. The transfer function of $\mathcal{B}^p$ is then given as the pth power:

$$\hat{b}^p = \cos^p(\pi\tilde{k}/2) = 1 - \frac{p}{8}(\pi\tilde{k})^2 + \left(\frac{3p^2 - 2p}{384}\right)(\pi\tilde{k})^4 + O(\tilde{k}^6). \quad (11.20)$$

The graphical representation of the transfer function in Fig. 11.1b reveals that binomial filters are much better smoothing filters than box

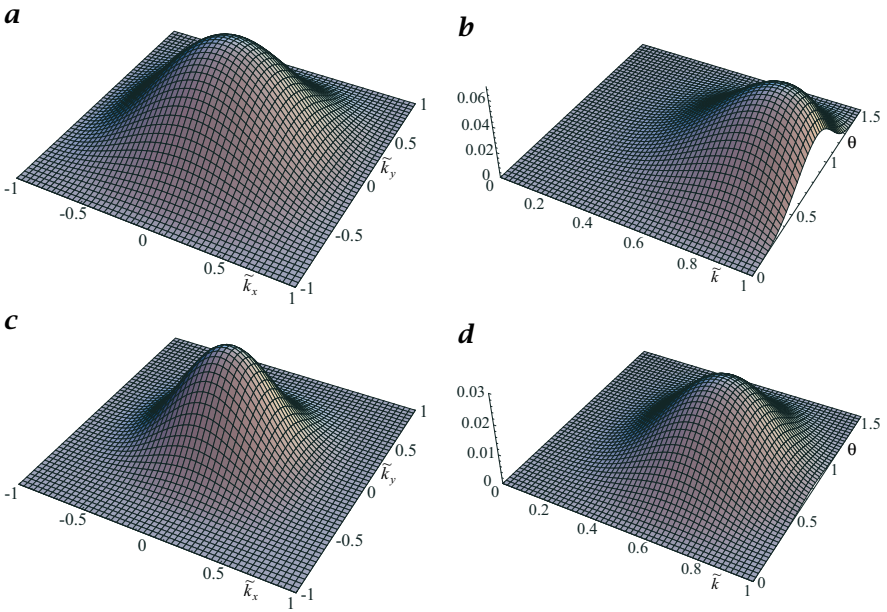

Figure 11.5: *Transfer function of two-dimensional binomial filters:* ***a*** $\mathcal{B}^2$*;* ***b*** *anisotropy* $\hat{B}^2(\tilde{k},\theta) - \hat{B}^2(\tilde{k},0)$ *in a* (k,θ) *diagram;* ***c*** $\mathcal{B}^4$*;* ***d*** *anisotropy for* $\mathcal{B}^4$ *as in* ***b****.*

filters. The transfer function decreases monotonically and approaches zero at the largest wave number. The smallest mask, $\mathcal{B}^2$, has a halfwidth of $\tilde{k}/2$. This is a periodic structure which is sampled four times per wavelength. For larger masks, both the transfer function and the filter masks approach the Gaussian distribution with an equivalent variance. Larger masks result in smaller half-width wave numbers in agreement with the uncertainty relation.

11.4.3 2-D Binomial Filter

Two-dimensional binomial filters can be composed from a horizontal and a vertical 1-D filter:

$$\mathcal{B}^p = \mathcal{B}_x^p \mathcal{B}_y^p. \tag{11.21}$$

The smallest mask of this kind is a 3×3-binomial filter ($p = 2$):

$$\boldsymbol{B}^2 = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}. \tag{11.22}$$

The transfer function of the 2-D binomial filter $\mathcal{B}^p$ with $(p+1) \times (p+1)$ coefficients is easily derived from the transfer functions of the 1-D filters

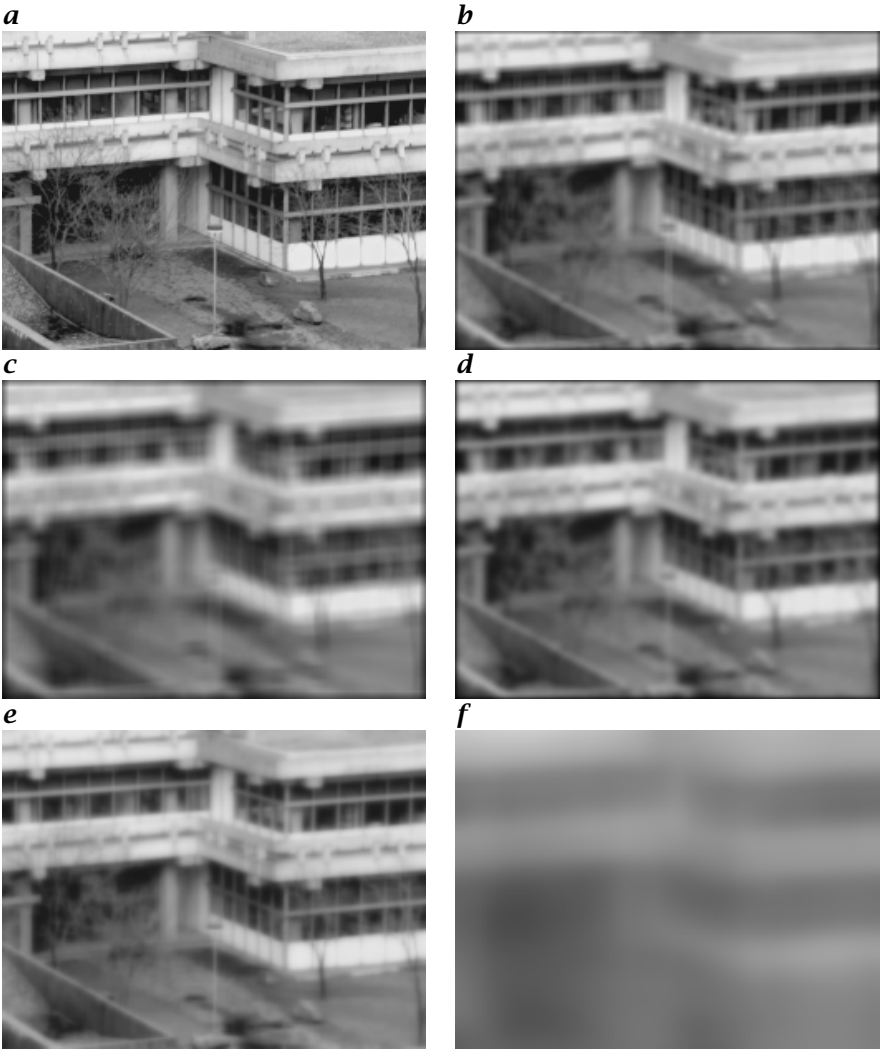

Figure 11.6: *Application of smoothing filters:* ***a*** *original image;* ***b*** 5×5 *box filter;* ***c*** 9×9 *box filter;* ***d*** 17×17 *binomial filter* ($\mathcal{B}^{16}$)*; a set of recursive filters Eq. (11.36) running in horizontal and vertical directions;* ***e*** $p = 2$*;* ***f*** $p = 16$.

Eq. (11.20) as

$$\hat{b}^p = \hat{b}_y^p \hat{b}_x^p = \cos^p(\pi \tilde{k}_y/2) \cos^p(\pi \tilde{k}_x/2), \tag{11.23}$$

and correspondingly for a 3-D filter as

$$\hat{b}^p = \hat{b}_z^p \hat{b}_y^p \hat{b}_x^p = \cos^p(\pi \tilde{k}_z/2) \cos^p(\pi \tilde{k}_y/2) \cos^p(\pi \tilde{k}_x/2). \tag{11.24}$$

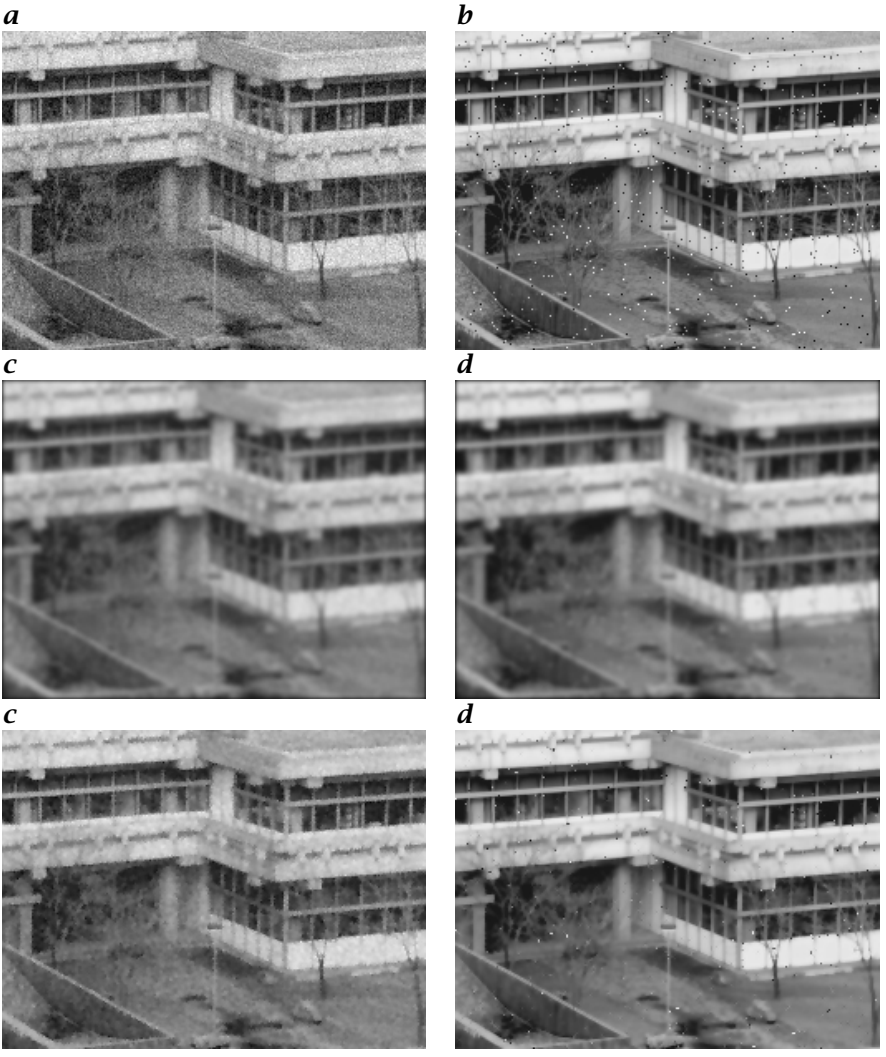

Figure 11.7: *Suppression of noise with smoothing filters:* ***a*** *image from Fig. 11.6a with Gaussian noise;* ***b*** *image with binary noise;* ***c*** *image* ***a*** *and* ***d*** *image* ***b*** *filtered with a* 9×9 *binomial filter* ($\mathcal{B}^8$)*;* ***e*** *image* ***a*** *and* ***f*** *image* ***b*** *filtered with a* 3×3 *median filter (Section 11.7.1).*

The transfer functions of $\mathcal{B}^2$ and $\mathcal{B}^4$ are shown in Fig. 11.5. Already the small 3×3 filter is remarkably isotropic. Larger deviations from the circular contour lines can only be recognized for larger wave numbers, when the transfer function has dropped to 0.3 (Fig. 11.5a). This property can be shown by expanding Eq. (11.23) in a Taylor series using cylindrical

coordinates $\tilde{\boldsymbol{k}} = [\tilde{k}, \theta]^T$:

$$\hat{b}^p \approx 1 - \frac{p}{8}(\pi\tilde{k})^2 + \frac{2p^2 - p}{256}(\pi\tilde{k})^4 - \frac{p\cos 4\theta}{768}(\pi\tilde{k})^4. \tag{11.25}$$

Only the second-order term is isotropic. In contrast, the fourth-order term contains an anisotropic part which increases the transfer function in the direction of the diagonals (Fig. 11.5a). A larger filter (larger p) is less anisotropic as the isotropic term with $\tilde{k}^4$ increases quadratically with p while the anisotropic term with $\tilde{k}^4 \cos 4\theta$ increases only linearly with p. Already the 5×5 filter (Fig. 11.5b) is remarkably isotropic. The insignificant anisotropy of the binomial filters also becomes apparent when applied to the test image in Fig. 11.4.

11.4.4 Evaluation

Figure 11.6b, c show smoothing with two different binomial filters. We observe that the edges get blurred. Fine structures as in the branches of the tree are lost. Smoothing suppresses noise. Binomial filters can reduce the noise level of zero-mean *Gaussian noise* (Section 3.4.2) considerably but only at the price of blurred details (Fig. 11.7a, c). *Binary noise* (also called *impulse noise*), which causes wrong gray values for a few randomly distributed pixels (Fig. 11.7b) (for instance due to transmission errors), is suppressed only poorly by linear filters. The images are blurred, but the error caused by the binary noise is not eliminated but only distributed.

11.4.5 Fast Computation

We close our consideration of binomial filters with some remarks on fast algorithms. A direct computation of a $(2p+1) \times (2p+1)$ filter mask requires $(2p+1)^2$ multiplications and $(2p+1)^2 - 1$ additions. If we decompose the binomial mask into elementary smoothing masks $1/2\,[1\ 1]$ and apply this mask in horizontal and vertical directions $2p$ times each, we only need $4p$ additions. All multiplications can be handled much more efficiently as shift operations. For example, the computation of a 17×17 binomial filter requires only 32 additions and some shift operations compared to 289 multiplications and 288 additions needed for the direct approach.

11.5 Filters as Networks‡

The binomial filters we have discussed in this section are built from the simplest elementary operations we can think of: scaling of pixels and the addition of neighboring pixels. For both of these operations we can construct a circuit element that performs the corresponding operation. Figure 11.8 shows a *scaler*,

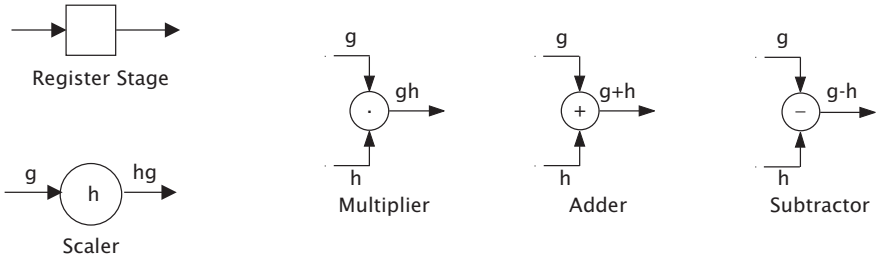

Figure 11.8: *Elementary circuits for performing discrete filter operations.*

an *adder*, a *subtractor*, a *multiplier*, and a *shift-register stage*. The circuit elements can perform the operations either analogously or digitally. With these circuit elements, we built FIR filters with electronic networks in an instructive way. This results also in a new perspective for filter operations.

As a first example, we consider the 1-D binomial mask $\boldsymbol{B}^4 = 1/16\,[1\ 4\ 6\ 4\ 1]$. Figure 11.9 shows different implementations for computing the filter output for one pixel. While direct implementations result in irregular-shaped circuit nets, the composition of the filter with the $\boldsymbol{B} = 1/2\,[1\ 1]$ mask gives a regular mesh of operations. For the calculation of a single output pixel, we need 10 additions, more than for the direct implementation. To calculate the filter output of the next pixel, we only need four additions if we store the intermediate results on each level of the filter net from the computations of the previous pixel (Fig. 11.9d).

Actually, we could build a net of these circuits, spanning the whole vector, to compute the binomial smoothing in parallel (Fig. 11.10). Such a net has a number of interesting properties. Each level of the net corresponds to a filtering of the image with the elementary smoothing mask $1/2\,[1\ 1]$. Thus we obtain not only the final result but all intermediate smoothing results. The grid points of the individual layers change from regular grid points to intermediate grid points in a natural way.

The network model for filter operations is also suitable for illustrating the different approaches in handling the boundary problems of filtering. We could close the net into a ring. This corresponds to a *cyclic convolution*. Or we could extend the net beyond the edges of the vector, so that we get all the nodes needed to calculate the first and last point. Then we can either fill the grid points in the lowest levels lying outside the vector with zeroes or we can extrapolate them in an appropriate manner from the points within the vector.

The extension of filter nets to two dimensions is straightforward for separable filters. The nets are then composed of nets alternately connecting the pixels in the horizontal or vertical direction. The filter nets are valuable tools for algorithm design. As we have seen, they are especially useful in making efficient use of intermediate results and in getting a clear idea of the boundary problems at the edges of images.

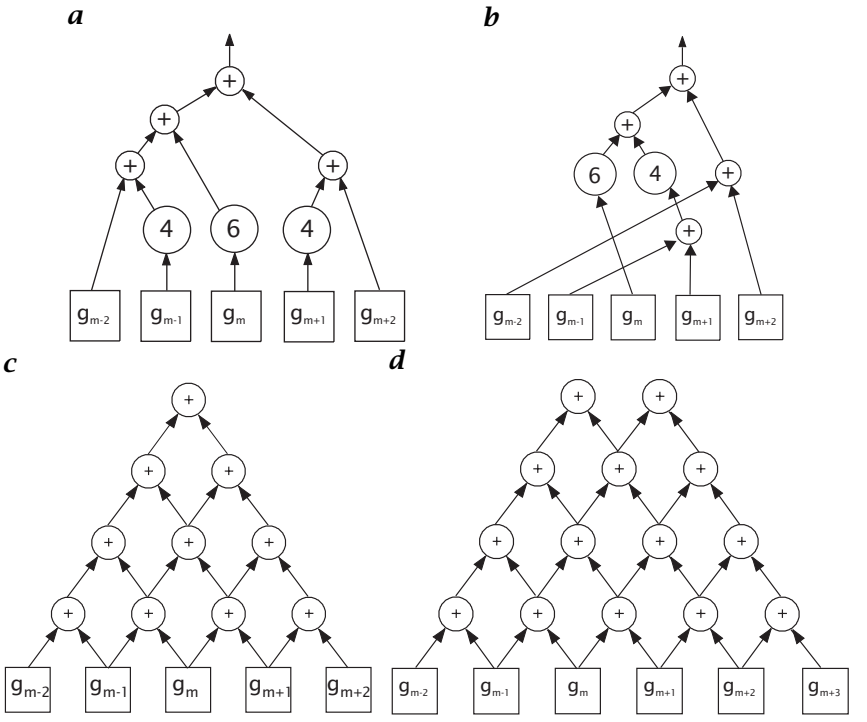

Figure 11.9: *Different circuit nets for performing the binomial smoothing filter operation* $\mathbf{B}^4 = 1/16\,[1\ 4\ 6\ 4\ 1]$*:* ***a*** *direct implementation;* ***b*** *saving multiplications;* ***c*** *composition with the elementary filter* $\mathbf{B} = 1/2\,[1\ 1]$*;* ***d*** *computation for the next pixel.*

11.6 Efficient Large-Scale Averaging‡

Despite the efficient implementation of binomial smoothing filters $\mathcal{B}^r$ by cascaded convolution with $\mathcal{B}$, the number of computations increases dramatically for smoothing masks with low cutoff wave numbers, because the standard deviation of the filters is proportional to the square root of p according to Eq. (3.42):

$$\sigma = \sqrt{p/4}. \tag{11.26}$$

Let us consider a smoothing operation over a circle with a radius of about only 1.73 pixels, corresponding to a variance $\sigma^2 = 3$. According to Eq. (11.26) we need to apply $\mathcal{B}^{12}$ which — even in an efficient separable implementation — requires 24 (36) additions and 2 (3) shift operations for each pixel in a 2-D (3-D) image. If we want to smooth over the double distance ($\sigma^2 = 12$, radius ≈ 3.5, $\mathcal{B}^{48}$) the number of additions quadruples to 96 (144) per pixel in 2-D (3-D) space.

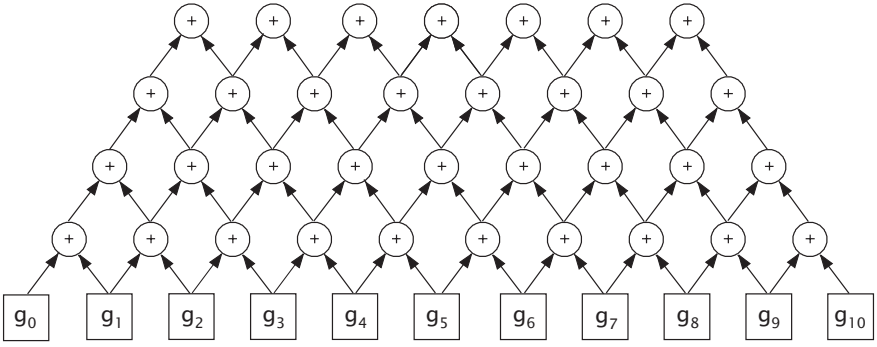

Figure 11.10: *Circuit net for computing the 1-D binomial smoothing filter $\mathcal{B}^4$.*

11.6.1 Multistep Averaging‡

The problem of slow large-scale averaging originates from the small distance between the pixels averaged in the elementary $\boldsymbol{B} = 1/2\,[1\ 1]$ mask. In order to overcome this problem, we may use the same elementary averaging process but with more distant pixels and increase the standard deviation for smoothing correspondingly. In two dimensions, the following masks could be applied along diagonals ($\sigma \cdot \sqrt{2}$):

$$\boldsymbol{B}_{x+y} = \frac{1}{4}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{B}_{x-y} = \frac{1}{4}\begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \tag{11.27}$$

or, with double step width along axes ($\sigma \cdot 2$) and in three dimensions,

$$\boldsymbol{B}_{2x} = \frac{1}{4}\,[1\ 0\ 2\ 0\ 1], \quad \boldsymbol{B}_{2y} = \frac{1}{4}\begin{bmatrix} 1 \\ 0 \\ 2 \\ 0 \\ 1 \end{bmatrix}, \quad \boldsymbol{B}_{2z} = \frac{1}{4}\begin{bmatrix} 1 \\ 0 \\ 2 \\ 0 \\ 1 \end{bmatrix}_z. \tag{11.28}$$

The subscripts in these masks denote the stepping width and coordinate direction. $\boldsymbol{B}_{x+y}$ averages the gray values at two neighboring pixels in the direction of the main diagonal. $\boldsymbol{B}_{2x}$ computes the mean of two pixels at a distance of 2 in the x direction. The standard deviation of these filters is proportional to the distance between the pixels. The most efficient implementations are multistep masks along the axes. They have the additional advantage that because of separability, the algorithms can be applied to image data of arbitrary dimensions.

The problem with these filters is that they perform a subsampling. Consequently, they are no longer filters for larger wave numbers. If we take, for example, the symmetric 2-D $\mathcal{B}^2_{2x}\mathcal{B}^2_{2y}$ filter, we effectively work on a grid with a doubled grid constant in the spatial domain. Hence, the reciprocal grid in the wave number space has half the grid width and the transfer function is periodically replicated once in both directions (Fig. 11.11). Generally, the zero lines

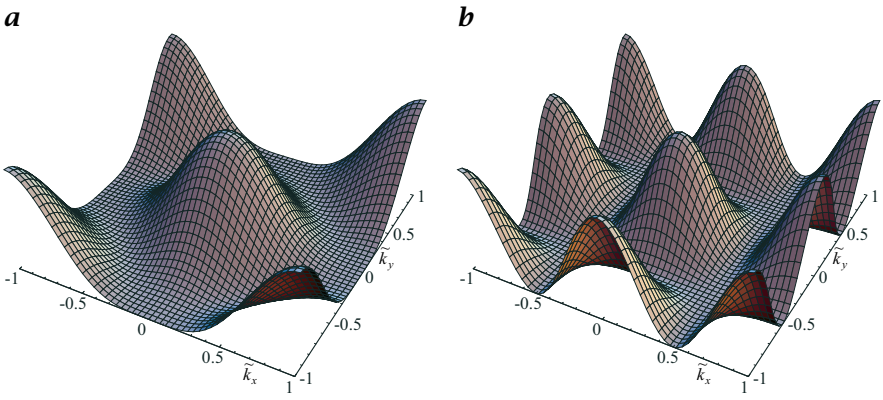

Figure 11.11: *Transfer function of the binomial mask applied* ***a*** *in the diagonal direction ($\mathcal{B}^2_{x+y}\mathcal{B}^2_{x-y}$) and* ***b*** *with double step width in axis directions ($\mathcal{B}^2_{2x}\mathcal{B}^2_{2y}$).*

of the transfer function of masks with larger step width reflect this reciprocal grid. For convolution with two neighboring pixels in the direction of the two diagonals, the reciprocal grid is turned by 45°. The grid constant of the reciprocal grid is a factor $\sqrt{2}$ smaller than that of the original grid.

Used individually, these filters are not of much help. But we can use them in cascade, starting with directly neighboring pixels. Then the zero lines of the transfer functions, which lie differently for each pixel distance, efficiently force the transfer function close to zero for large wave number ranges.

Cascaded multistep binomial filtering leads to a significant performance increase for large-scale smoothing. For normal separable binomial filtering, the number of computations is proportional to $\sigma^2(O(\sigma^2))$. For multistep binomial filtering it depends only logarithmically on σ $(O(\mathrm{ld}\sigma^2))$ if a cascade of filter operations with recursive step width doubling is performed:

$$\underbrace{\mathcal{B}^p_{2^{s-1}x}\cdots\mathcal{B}^p_{8x}\mathcal{B}^p_{4x}\mathcal{B}^p_{2x}\mathcal{B}^p_{x}}_{s\text{ times}}. \tag{11.29}$$

Such a mask has the standard deviation

$$\sigma^2 = \underbrace{p/4 + p + 4p + \ldots + 4^{s-1}p}_{s\text{ times}} = \frac{p}{12}(4^s - 1) \tag{11.30}$$

and the transfer function

$$\prod_{s'=0}^{s-1}\cos^p(2^{s'-1}\pi\tilde{k}). \tag{11.31}$$

Thus, for s steps only ps additions are required while the standard deviation grows exponentially with $\approx\sqrt{p/12}\cdot 2^s$.

With the parameter p, we can adjust the degree of isotropy and the degree of residual inhomogeneities in the transfer function. A very efficient implementation is given by using p = 2 ($\boldsymbol{B}^2$ = 1/4[1 2 1] in each direction). However the

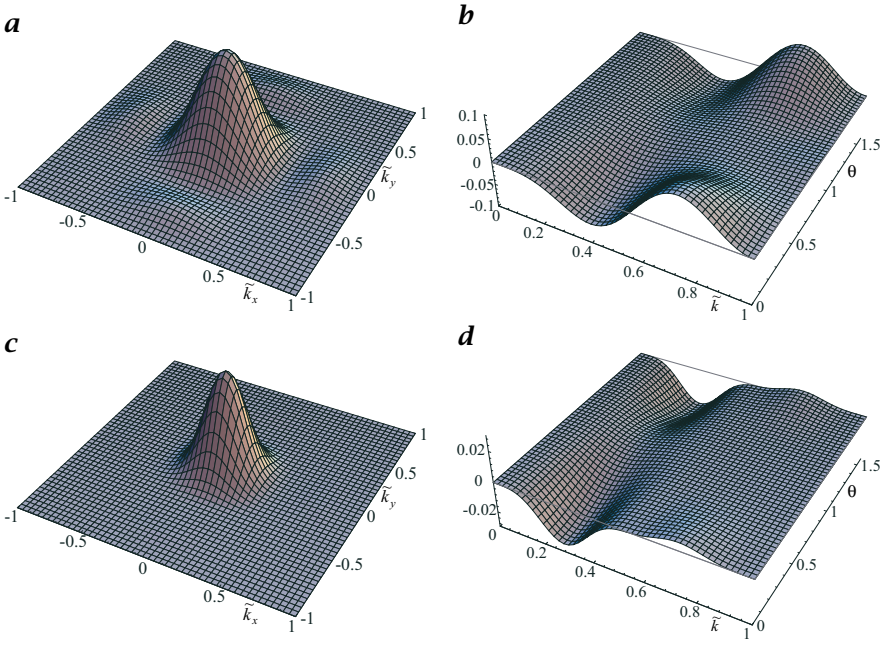

Figure 11.12: *Transfer function of cascaded multistep binomial filters and their anisotropy:* ***a*** $\mathcal{B}_2^2\mathcal{B}_1^2$, ***b*** $\hat{B}_2^2\hat{B}_1^2(\tilde{k},\theta)-\hat{B}_2^2\hat{B}_1^2(\tilde{k},0)$, ***c*** $\mathcal{B}_2^4\mathcal{B}_1^4$, ***d*** $\hat{B}_2^4\hat{B}_1^4(\tilde{k},\theta)-\hat{B}_2^4\hat{B}_1^2(\tilde{k},0)$. *The anisotropy is shown in polar coordinates* $(\tilde{k},\theta)$ *as the deviation from the transfer function in the* x *direction.*

residual side peaks at high wave numbers with maximal amplitudes up to 0.08 are still significant disturbances (Fig. 11.12a, b, Fig. 11.13a, b).

With the next larger odd-sized masks (p = 4, $\boldsymbol{B}^4$ = 1/16[1 4 6 4 1] in each direction) these residual side peaks at high wave numbers are suppressed well below 0.005 (Fig. 11.12c, d, Fig. 11.13c, d). This is about the relative resolution of 8-bit images and should therefore be sufficient for most applications. With still larger masks, they could be suppressed even further. Figure 11.14 shows the first four steps of multistep averaging with the $\boldsymbol{B}^4$ mask, illustrating how quickly the smoothing reaches large scales.

11.6.2 Multigrid Averaging‡

Multistep cascaded averaging can be further enhanced by converting it into a multiresolution technique. The idea of multigrid smoothing is very simple. When a larger-step mask is involved, this operation can be applied on a correspondingly coarser grid. This means that the last operation before using the larger-step mask needs to compute the convolution only at the grid points used by the following coarser grid operator. This sampling procedure is denoted by a special syntax in the operator index. $\mathcal{O}_{x|2}$ means: Apply the operator in the x direction and advance the mask two pixels in the x direction. Thus, the output of the filter operator has only half as many pixels in the x direction as the input.

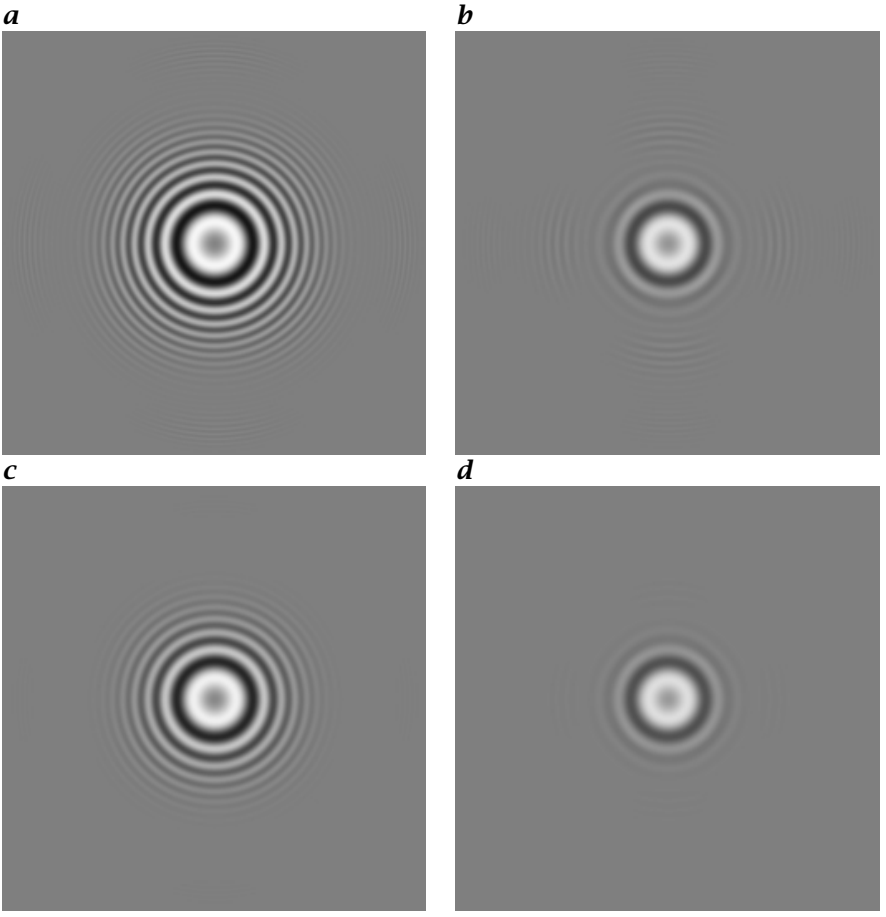

Figure 11.13: *Cascaded multistep averaging with step width doubling according to Eq. (11.29), applied to the ring test pattern:* ***a*** $\mathcal{B}_2^2\mathcal{B}_1^2$, ***b*** $\mathcal{B}_4^2\mathcal{B}_2^2\mathcal{B}_1^2$, ***c*** $\mathcal{B}_2^4\mathcal{B}_1^4$, *and* ***d*** $\mathcal{B}_4^4\mathcal{B}_2^4\mathcal{B}_1^4$.

Multigrid smoothing makes the number of computations essentially independent of the standard deviation of the smoothing mask. We again consider a sequence of 1-D binomial filters:

$$\underbrace{\mathcal{B}_{x\downarrow 2}^p \cdots \mathcal{B}_{x\downarrow 2}^p \mathcal{B}_{x\downarrow 2}^p}_{s \text{ times}}.$$

Since $\mathcal{B}_{x|2}^p$ takes p operations, the operator sequence takes

$$p \sum_{s'=1}^{s} \frac{1}{2^{s'-1}} = 2p\left(1 - \frac{1}{2^{s-1}}\right) < 2p.$$

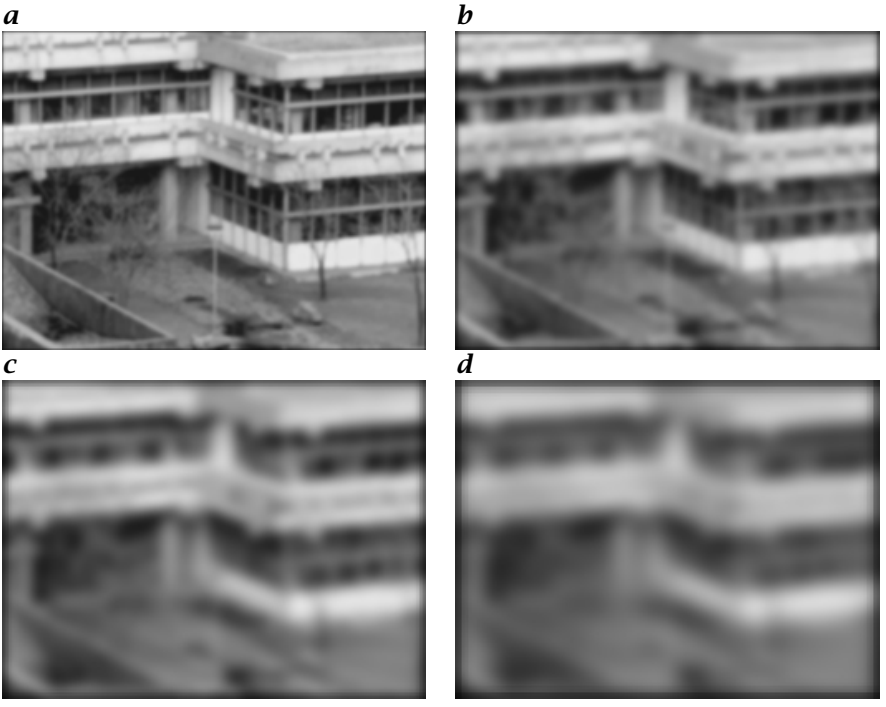

Figure 11.14: *Cascaded multistep averaging with step width doubling according to Eq. (11.29), applied to image Fig. 11.6a with* ***a*** *one,* ***b*** *two,* ***c*** *three, and* ***d*** *four steps using the* $\boldsymbol{B}^4$ *filter.*

As for the multistep approach, Eq. (11.30), the standard deviation of the operator sequence is

$$\sigma^2 = \frac{p}{12}(4^s - 1). \tag{11.32}$$

Thus, smoothing to any degree takes not more than twice as many operations as smoothing at the first step! As for multistep binomial filters, the standard deviation grows by a factor of two. Also — as long as $\hat{\boldsymbol{B}}^p(\tilde{k}) = 0 \quad \forall \tilde{k} \geq 1/2$ — the transfer functions of the filters are the same as for the multistep filters.

11.6.3 Recursive Averaging‡

A totally different approach to large-scale averaging is given by recursive filtering introduced in Section 4.3. The recursion essentially gives a convolution filter an infinite point spread function. The basic advantage of recursive filters is that they can easily be "tuned", as we have demonstrated with the simple lowpass filter in Section 4.3.5. In this section, the focus is on the design of averaging filters that meet the criteria we discussed earlier in Section 11.2, especially the zero-shift property (Section 11.2.1) that is not met by causal recursive filters.

Basically, recursive filters work the same as non-recursive filters. In principle, we can replace any recursive filter with a non-recursive filter whose filter mask is identical to the point spread function of the recursive filter. The real problem is the design of the recursive filter, i. e., the determination of the filter coefficients for a desired transfer function.

While the theory of one-dimensional recursive filters is standard knowledge in digital signal processing (see, for example, Oppenheim and Schafer [133]), the design of two-dimensional filters is still not adequately understood. The main reason is the fundamental difference between the mathematics of one- and higher-dimensional z-transforms and polynomials [112].

Despite these theoretical problems, recursive filters can be applied successfully in digital image processing. In order to avoid the filter design problems, we will use only very simple recursive filters which are easily understood and compose them to more complex filters, similar to the way we constructed the class of binomial filters from the elementary smoothing mask $1/2\,[1\ 1]$. In this way we will obtain a class of recursive filters that may not be optimal from the point of view of filter design but are useful in practical applications.

In the first composition step, we combine causal recursive filters to symmetric filters. We start with a general one-dimensional recursive filter with the transfer function

$$^{+}\hat{\boldsymbol{A}} = a(\tilde{k}) + \mathrm{i}b(\tilde{k}). \tag{11.33}$$

The index + denotes the run direction of the filter in the positive coordinate direction. The transfer function of the same filter but running in the opposite direction is

$$^{-}\hat{\boldsymbol{A}} = a(\tilde{k}) - \mathrm{i}b(\tilde{k}). \tag{11.34}$$

Only the sign of the imaginary part of the transfer function changes, as it corresponds to the odd part of the point spread function, while the real part corresponds to the even part.

We now have two possible ways to combine the forward and backward running filters into symmetric filters useful for averaging:

$$\begin{array}{lll} \text{addition} & \hat{\boldsymbol{A}} = \frac{1}{2}\left[^{+}\hat{\boldsymbol{A}} + {}^{-}\hat{\boldsymbol{A}}\right] & = a(\tilde{k}) \\ \text{multiplication} & \hat{\boldsymbol{A}} = {}^{+}\hat{\boldsymbol{A}}\,{}^{-}\hat{\boldsymbol{A}} & = a^2(\tilde{k}) + b^2(\tilde{k}). \end{array} \tag{11.35}$$

Both techniques yield real transfer functions and thus even filters with zero shift that are suitable for averaging.

As the elementary recursive smoothing filter, we use the two-element lowpass filter we have already studied in Section 4.3.5:

$$^{\pm}\mathcal{A}_x:\ G'_{mn} = G'_{m,n\mp1} + \alpha(G_{mn} - G'_{m,n\mp1}) \quad \text{with} \quad 0 \le \alpha \le 1 \tag{11.36}$$

with the impulse response

$$(^{\pm}A_x)_{m,n} = \begin{cases} \alpha(1-\alpha)^n & n > 0, m = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{11.37}$$

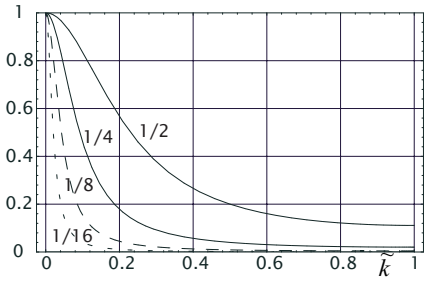

Figure 11.15: *Transfer function of the recursive lowpass filter Eq. (11.39) for different values of α = 1/2, 1/4, 1/8, and 1/16.*

The transfer function of this filter can easily be calculated by taking into account that the Fourier transform of Eq. (11.37) forms a *geometric series*:

$$^{\pm}\hat{A}_x(\tilde{k}) = \frac{\alpha}{1-(1-\alpha)\exp(\mp i\pi\tilde{k})}. \tag{11.38}$$

This relation is valid only approximately, since we broke off the infinite sum in Eq. (11.37) at $n = N-1$ because of the limited size of the image.

Consecutive filtering with a left and right running filter corresponds to a multiplication of the transfer functions

$$\hat{A}_x(\tilde{k}) = {}^{+}\hat{A}_x(\tilde{k})\,{}^{-}\hat{A}_x(\tilde{k}) \approx \frac{\alpha^2}{\alpha^2 + 2(1-\alpha)(1-\cos(\pi\tilde{k}))}. \tag{11.39}$$

The transfer function shows the characteristics expected for a lowpass filter (Fig. 11.15). At $\tilde{k} = 0$, $\hat{A}_x(\tilde{k}) = 1$; for small $\tilde{k}$, the transfer function falls off in proportion to $\tilde{k}^2$,

$$\hat{A}_x \approx 1 - \frac{1-\alpha}{\alpha^2}(\pi\tilde{k})^2 \quad \tilde{k} \ll 1, \tag{11.40}$$

and has a half-value wave number $\tilde{k}_c$ ($\hat{A}_x(\tilde{k}_c) = 1/2$) of

$$\tilde{k}_c \approx \frac{1}{\pi}\arcsin\frac{\alpha}{\sqrt{2(1-\alpha)}} \approx \frac{\alpha}{\sqrt{2}\pi}, \tag{11.41}$$

where the last approximation is only valid for $\alpha << 1$. At the highest wave number, $\tilde{k} = 1$, the transfer function has dropped off to

$$\hat{A}_x(1) \approx \frac{\alpha^2}{4(1-\alpha)+\alpha^2}. \tag{11.42}$$

In contrast to binomial filters, it is not exactly zero, but sufficiently small even for small values of α (Fig. 11.15).

Two-dimensional filters can be composed from one-dimensional filters running in the horizontal and vertical directions:

$$\mathcal{A} = \mathcal{A}_x\mathcal{A}_y = {}^{+}\mathcal{A}_x\,{}^{-}\mathcal{A}_x\,{}^{+}\mathcal{A}_y\,{}^{-}\mathcal{A}_y. \tag{11.43}$$

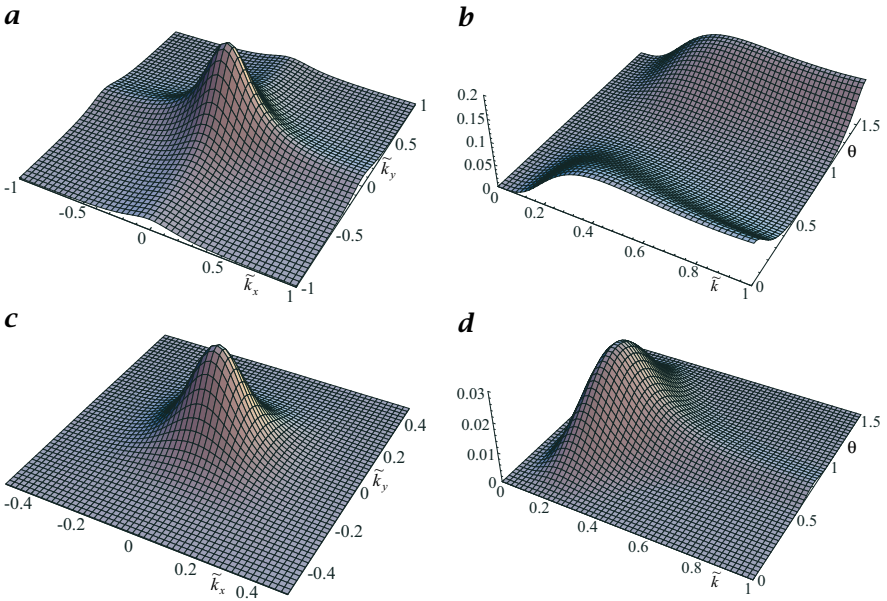

Figure 11.16: *Transfer functions of two-dimensional recursive lowpass filters:* ***a*** $\mathcal{A}$ *with* $\alpha = 1/2$, ***b*** *anisotropy of* ***a****:* $\hat{A}(k,\theta) - \hat{A}(k,\pi/4)$, ***c*** $\mathcal{A}'$ *with* $\alpha = 1/2$, *and* ***d*** *anisotropy of* ***c****:* $\hat{A}'(k,\theta) - \hat{A}'(k,0)$.

This filter (Fig. 11.16) is less isotropic than binomial filters (Fig. 11.5). Averaging in coordinate directions is considerably less than in the other directions. However, recursive filters have the big advantage that the computational effort does not depend on the degree of averaging. With the simple first-order recursive filter, we can select the degree of averaging with an appropriate choice of the filter parameter α (Eq. (11.41)). The isotropy of recursive filters can be further improved by running additional filters along the diagonals:

$$\mathcal{A}' = \mathcal{A}_x \mathcal{A}_y \mathcal{A}_{x-y} \mathcal{A}_{x+y}. \tag{11.44}$$

The subscripts $x-y$ and $x+y$ denote the main and second diagonal, respectively. The transfer function of such a filter is shown in Fig. 11.16b.

In contrast to non-recursive filters, the computational effort does not depend on the cut-off wave number. If $\alpha = 2^{-l}$ in Eq. (11.36), the filter can be computed without any multiplication:

$$G'_{mn} = \left[G'_{m,n\pm1} \cdot 2^l - G'_{m,n\pm1} + G_{mn}\right] \cdot 2^{-l}, \quad l > 1. \tag{11.45}$$

The two-dimensional filter $\mathcal{A}$ needs only 8 additions and shift operations per pixel, while the $\mathcal{A}'$ filter, running in 4 directions, needs twice as many operations. This is not more efficient than the multigrid approach with binomial masks (Section 11.6.2) that makes a much better isotropic filter.

11.7 Nonlinear Averaging

The linear averaging filters discussed so far blur edges. Even worse, if the mask of the smoothing operator crosses an object edge it contains pixels from both the object and the background, giving a meaningless result from the filter. The same is true if averages are performed when a certain number of pixels in an image show erroneous values, e. g., because of a transmission error. The question, therefore, is whether it is possible to perform an averaging that does not cross object boundaries or that ignores certain pixels. Such a procedure can only be applied, of course, if we have already detected the edges or any distorted pixel.

In this section, we discuss three types of nonlinear averaging filter: the classical median filter (Section 11.7.1); weighted averaging, also known as normalized convolution (Section 11.7.2); and steerable averaging (Section 11.7.3), where we control the direction and/or degree of averaging with the local content of the neighborhood.

11.7.1 Median Filter

Linear filters effectively suppress Gaussian noise but perform very poorly in case of binary noise (Fig. 11.7). Using linear filters that weigh and sum up, we assume that each pixel carries some useful information. Pixels distorted by transmission errors have lost their original gray value. Linear smoothing does not eliminate this information but carries it on to neighboring pixels. Thus the appropriate operation to process such distortions is to detect these pixels and to eliminate them.

This is exactly what a *rank-value filter* does (Section 4.4). The pixels within the mask are sorted and one pixel is selected. In particular, the *median filter* selects the medium value. As binary noise completely changes the gray value, it is very unlikely that it will show the medium gray value in the neighborhood. In this way, the medium gray value of the neighborhood is used to restore the gray value of the distorted pixel.

The following examples illustrate the effect of a 1×3 median filter $\mathcal{M}$:

$$\mathcal{M}[\cdots\ 1\ 2\ 3\ 7\ 8\ 9\ \cdots] = [\cdots\ 1\ 2\ 3\ 7\ 8\ 9\ \cdots]$$

$$\mathcal{M}[\cdots\ 1\ 2\ 102\ 4\ 5\ 6\ \cdots] = [\cdots\ 1\ 2\ 4\ 5\ 5\ 6\ \cdots]$$

$$\mathcal{M}[\cdots\ 0\ 0\ 0\ 9\ 9\ 9\ \cdots] = [\cdots\ 0\ 0\ 0\ 9\ 9\ 9\ \cdots]$$

As expected, the median filter eliminates runaways. The two other gray value structures — a monotonically increasing ramp and an edge between two plateaus of constant gray value — are preserved. In this way a median filter effectively eliminates binary noise without significantly blurring the image (Fig. 11.7e). Gaussian noise is less effectively eliminated (Fig. 11.7f).

The most important deterministic properties of a one-dimensional $2N + 1$ median filter can be formulated using the following definitions.

- A *constant neighborhood* is an area with $N + 1$ equal gray values.
- An *edge* is a monotonically increasing or decreasing area between two constant neighborhoods.
- An *impulse* is an area of at most N points surrounded by constant neighborhoods with the same gray value.
- A *root* or *fix point* is a signal that is preserved under the median filter operation.

With these definitions, the deterministic properties of a median filter can be described very compactly:

- Constant neighborhoods and edges are fix points.
- Impulses are eliminated.

Iterative filtering of an image with a median filter results in an image containing only constant neighborhoods and edges. If only single pixels are distorted, a 3×3 median filter is sufficient to eliminate them. If clusters of distorted pixels occur, larger median filters must be used.

The statistical properties of the median filter can be illustrated with an image containing only constant neighborhoods, edges, and impulses. The power spectrum of impulses is flat (*white noise*). As the median filter eliminates impulses, the power spectrum decreases homogeneously. The contribution of the edges to a certain wave number is not removed. This example also underlines the nonlinear nature of the median filter.

11.7.2 Weighted Averaging

In Section 3.1, we saw that gray values at pixels, just like any other experimental data, may be characterized by individual errors that have to be considered in any further processing. As an introduction, we first discuss the averaging of a set of N data g_n with standard deviations σ_n. From elementary statistics, it is known that appropriate averaging requires the weighting of each data point g_n with the inverse of the variance $w_n = 1/\sigma_n^2$. Then, an estimate of the mean value is given by

$$\overline{g} = \sum_{n=1}^{N} g_n/\sigma_n^2 \Bigg/ \sum_{n=1}^{N} 1/\sigma_n^2 \tag{11.46}$$

while the standard deviation of the mean is

$$\sigma_{\overline{g}}^2 = 1 \Bigg/ \sum_{n=1}^{N} 1/\sigma_n^2 \,. \tag{11.47}$$

The weight of an individual data point in Eq. (11.46) for computation of the mean is the higher, the lower its statistical error.

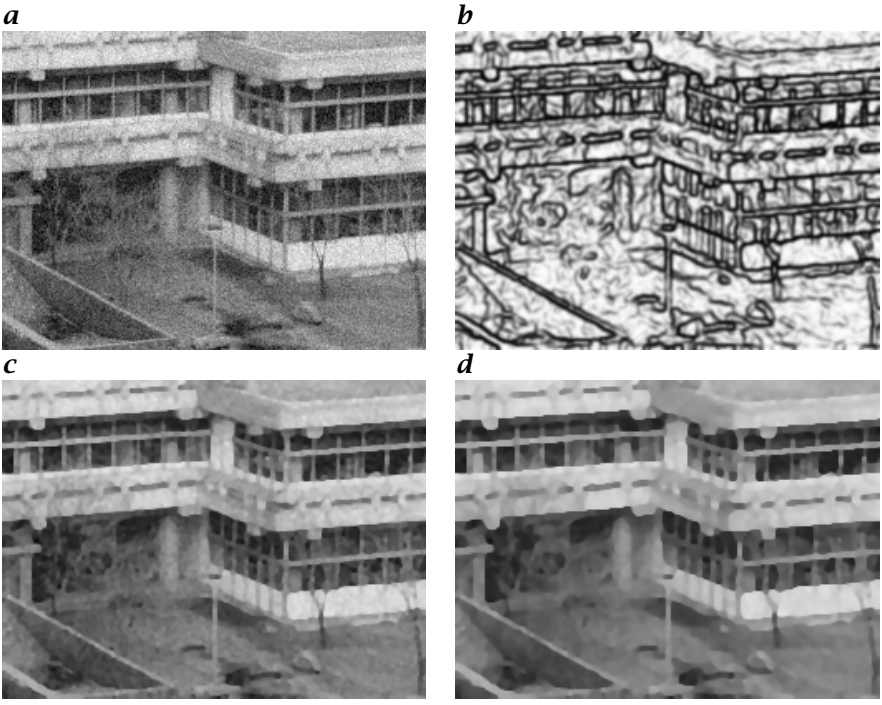

Figure 11.17: *Weighted averaging using the edge strength to hinder smoothing at edges:* ***a*** *image from Fig. 11.6a with added Gaussian noise;* ***b*** *weighting image after 5 convolutions; image after* ***c*** *two and* ***d*** *five normalized convolutions using a* $\mathcal{B}^2$ *binomial smoothing mask (compare with Fig. 11.7).*

The application of *weighted averaging* to image processing is known as *normalized convolution* [57]. The averaging is now extended to a local neighborhood. Each pixel enters the convolution sum with a weighting factor associated with it. Thus, normalized convolution requires two images. One is the image to be processed, the other an image with the weighting factors.

By analogy to Eqs. (11.46) and (11.47), normalized convolution is defined by

$$\boldsymbol{G}' = \frac{\boldsymbol{H} * (\boldsymbol{W} \cdot \boldsymbol{G})}{\boldsymbol{H} * \boldsymbol{W}}, \tag{11.48}$$

where $\boldsymbol{H}$ is any convolution mask, $\boldsymbol{G}$ the image to be processed, and $\boldsymbol{W}$ the image with the weighting factors. A normalized convolution with the mask $\boldsymbol{H}$ essentially transforms the set of the image $\boldsymbol{G}$ and the weighting image $\boldsymbol{W}$ into a new image $\boldsymbol{G}'$ and a new weighting image $\boldsymbol{W}' = \boldsymbol{H} * \boldsymbol{W}$ which can undergo further processing.

In this sense, normalized convolution is nothing complicated or special. It is just adequate consideration of pixels with spatially variable

statistical errors. "Standard" convolution can be regarded as a special case of normalized convolution. Then all pixels are assigned the same weighting factor and it is not required to use a weighting image, since the factor remains a constant.

The flexibility of normalized convolution is given by the choice of the weighting image. The weighting image does not necessarily get associated with an error. It can be used to select and/or amplify pixels with certain features. In this way, normalized convolution becomes a versatile nonlinear operator.

As an example, Fig. 11.17 shows an noisy image that is filtered by normalized convolution using an weighting image that hinders smoothing at edges.

11.7.3 Steerable Averaging

The idea of *steerable filters* is to make the convolution mask dependent on the local image structure. This is a general concept which is not restricted to averaging but can be applied to any type of convolution process. The basic idea of steerable filters is as follows. A steerable filter has some freely adjustable parameters that control the filtering. These could be various properties such as the degree of smoothing, the direction of smoothing, or both. It is easy to write down a filter mask with adjustable parameters. We have done this already for recursive filters in Eq. (11.36) where the parameter α determines the degree of smoothing. However, it is not computationally efficient to convolve an image with masks that are different at every pixel. Then, advantage can no longer be taken of the fact that masks are separable.

An alternative approach is to seek a base of a few filters, and to use these filters to compute a set of filtered images. Then, these images are interpolated using adjustable parameters. In operator notation this reads

$$\mathcal{H}(\alpha) = \sum_{p=1}^{P} f_p(\alpha)\mathcal{H}_p, \tag{11.49}$$

where $\mathcal{H}_p$ is the pth filter and $f_p(\alpha)$ a scalar interpolation function of the steering parameter α. Two problems must be solved when using steerable filters. First, and most basically, it is not clear that such a filter base H_p exists at all. Second, the relation between the steering parameter(s) α and the interpolation coefficients f_p must be found. If the first problem is solved, we mostly get the solution to the second for free.

As an example, a directional smoothing filter is to be constructed with the following transfer function:

$$\hat{h}_{\theta_0}(k,\theta) = 1 - f(k)\cos^2(\theta - \theta_0). \tag{11.50}$$

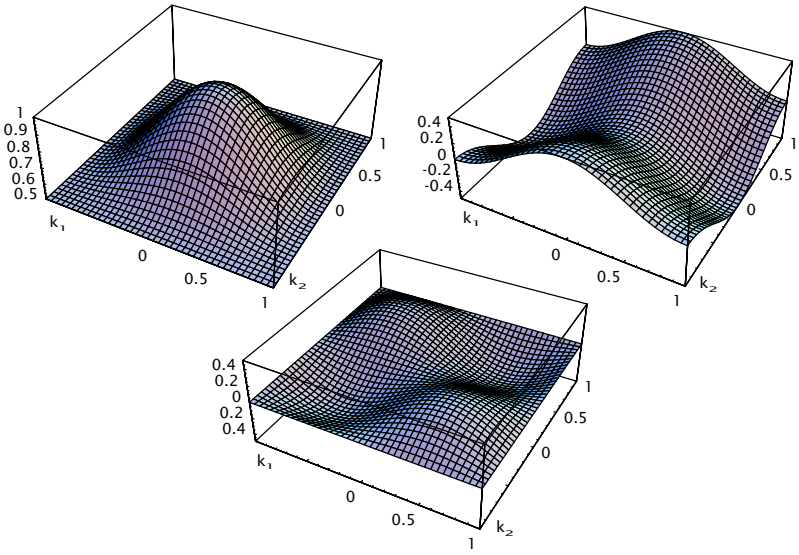

Figure 11.18: *Transfer functions for the three base filters for directional smoothing according to Eq. (11.54).*

In this equation, cylindrical coordinates (k, θ) are used in the Fourier domain. The filter in Eq. (11.50) is a *polar separable* filter with an arbitrary radial function $f(k)$. This radial component provides an arbitrary isotropic smoothing filtering.

The steerable angular term is given by $\cos^2(\theta - \theta_0)$. Structures oriented in the direction θ_0 remain in the image, while those perpendicular to θ_0 are completely filtered out. The angular width of the directional smoothing filter is $\pm 45^\circ$.

We separate the cosine function in Eq. (11.50) into trigonometric functions that depend only on either θ or the steering angle θ_0 and obtain

$$\hat{h}_{\theta_0}(k, \theta) = 1 - \frac{1}{2} f(k) \left[1 + \cos(2\theta_0)\cos(2\theta) + \sin(2\theta_0)\sin(2\theta)\right] \tag{11.51}$$

with the base filters

$$\hat{h}_1 = 1 - \frac{1}{2} f(k), \quad \hat{h}_2 = -\frac{1}{2} f(k)\cos(2\theta), \quad \hat{h}_3 = -\frac{1}{2} f(k)\sin(2\theta) \tag{11.52}$$

and the interpolation functions

$$f_1(\theta_0) = 1, \quad f_2(\theta_0) = \cos(2\theta_0), \quad f_3(\theta_0) = \sin(2\theta_0). \tag{11.53}$$

Thus three base filters are required. The filter $\hat{h}_1$ is an isotropic smoothing filter, the other two are directional filters. Although the equations

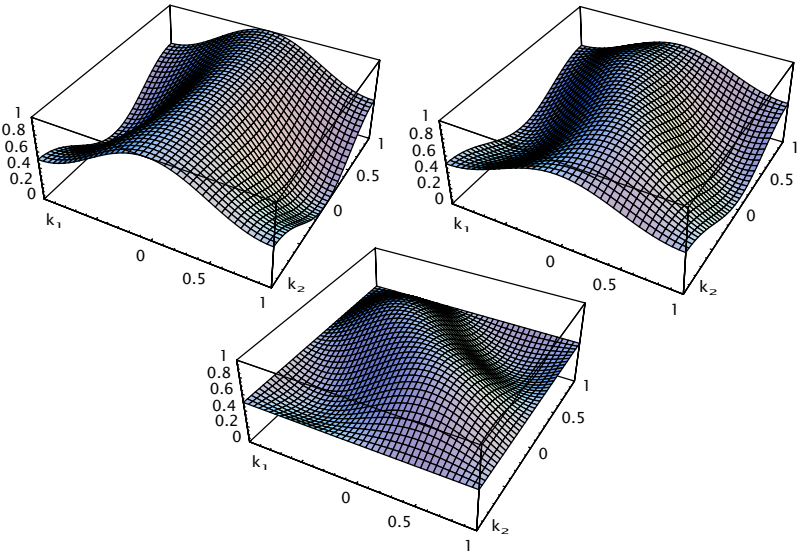

Figure 11.19: *Transfer functions for the steerable smoothing filter according to Eq. (11.51) using the base filters Eq. (11.54): smoothing in 0°, 22.5°, and 45°to the x axis.*

for this family of steerable directional smoothing filter are simple, it is not easy to implement polar separable base filters because they are not Cartesian separable and, thus, require careful optimization.

Nevertheless, it is even possible to implement this steerable smoothing filter with 3×3 base filters (Fig. 11.18). Because of symmetry properties of the transfer functions, we have not much choice to chose the filter coefficients and end up with the following three base filters:

$$\boldsymbol{H}_1 = \frac{1}{32}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 20 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \quad \boldsymbol{H}_2 = \frac{1}{32}\begin{bmatrix} 0 & -4 & 0 \\ 4 & 0 & 4 \\ 0 & -4 & 0 \end{bmatrix}, \quad \boldsymbol{H}_3 = \frac{1}{32}\begin{bmatrix} -2 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & -2 \end{bmatrix}$$

$$\begin{aligned} \hat{h}_1 &= \frac{1}{2} + \frac{1}{2}\cos^2(\pi\tilde{k}_1/2)\cos^2(\pi\tilde{k}_2/2) && \approx 1 - \frac{\pi^2\tilde{k}^2}{8}, \\ \hat{h}_2 &= \frac{1}{4}\left(\cos(\pi\tilde{k}_1) - \cos(\pi\tilde{k}_2)\right) && \approx \frac{\pi^2\tilde{k}^2}{8}\cos(2\theta), \\ \hat{h}_3 &= \frac{1}{8}\left(\cos(\pi(\tilde{k}_1+\tilde{k}_2)) - \cos(\pi(\tilde{k}_1-\tilde{k}_2))\right) && \approx \frac{\pi^2\tilde{k}^2}{8}\sin(2\theta). \end{aligned} \tag{11.54}$$

From Fig. 11.19 it is obvious that this simple implementation works well up to moderate wave numbers. At high wave number ($\tilde{k} > 0.5$), the directional filter does no longer work very well.

11.8 Averaging in Multichannel Images‡

At first glance, it appears that there is not much special about averaging of multichannel images: just apply the smoothing mask to each of the P channels individually:

$$\mathbf{G}' = \begin{bmatrix} \boldsymbol{G}'_1 \\ \boldsymbol{G}'_2 \\ \vdots \\ \boldsymbol{G}'_p \end{bmatrix} = \boldsymbol{H} * \mathbf{G} = \begin{bmatrix} \boldsymbol{H} * \boldsymbol{G}_1 \\ \boldsymbol{H} * \boldsymbol{G}_2 \\ \vdots \\ \boldsymbol{H} * \boldsymbol{G}_p \end{bmatrix}. \tag{11.55}$$

This simple concept can also be extended to normalized convolution (Section 11.7.2). If the same smoothing kernel is applied to all components, it is sufficient to use one common weighting image that can be appended as the $(P+1)$th component of the multicomponent image.

$$\begin{bmatrix} \boldsymbol{G}'_1 \\ \boldsymbol{G}'_2 \\ \vdots \\ \boldsymbol{G}'_P \\ \boldsymbol{W}' \end{bmatrix} = \begin{bmatrix} (\boldsymbol{H} * (\boldsymbol{W} \cdot \boldsymbol{G}_1))/(\boldsymbol{H} * \boldsymbol{W}) \\ (\boldsymbol{H} * (\boldsymbol{W} \cdot \boldsymbol{G}_2))/(\boldsymbol{H} * \boldsymbol{W}) \\ \vdots \\ (\boldsymbol{H} * (\boldsymbol{W} \cdot \boldsymbol{G}_P))/(\boldsymbol{H} * \boldsymbol{W}) \\ \boldsymbol{H} * \boldsymbol{W} \end{bmatrix} \tag{11.56}$$

A special case of multicomponent images is given when they represent features that can be mapped to angular coordinates. Typically, such features include the direction of an edge or the phase of a periodic signal. Features of this kind are cyclic and cannot be represented well as Cartesian coordinates.

Also, they cannot be averaged in this representation. Imagine an angle of +175° and –179°. The mean angle is 178°, since –179° = 360° – 179° = 181° is close to 175° and not (175° – 179°) / 2 = –2°.

Circular features such as angles are, therefore, better represented as unit vectors in the form $\bar{\boldsymbol{n}}_\theta = [\cos\theta, \sin\theta]^T$.

In this representation, they can be averaged correctly as illustrated in Fig. 11.20. The average vector points to the correct direction but its magnitude is generally smaller than 1:

$$(\bar{\boldsymbol{n}}_{\theta_1} + \bar{\boldsymbol{n}}_{\theta_2})/2 = \begin{bmatrix} \cos[(\theta_1 + \theta_2)/2] \\ \sin[(\theta_1 + \theta_2)/2] \end{bmatrix} \cos[(\theta_2 - \theta_1)/2]. \tag{11.57}$$

For an angle difference of 180°, the average vector has zero magnitude. The decrease of the magnitude of the average vector has an intuitive interpretation. The larger the scatter of the angle is, the less is the certainty of the average value. Indeed, if all directions are equally probable, the sum vector vanishes, while it grows in length when the scatter is low.

These considerations can be extended to the averaging of circular features. To this end we set the magnitude of the vector equal to the certainty of the quantity that is represented by the angle of the vector. In this way, short vectors add little and long vectors add more to the averaging procedure.

This is a very attractive form of weighted convolution since in contrast to normalized convolution (Section 11.7.2) it requires no time-consuming division. Of course, it works only with features that can be mapped adequately to an angle.

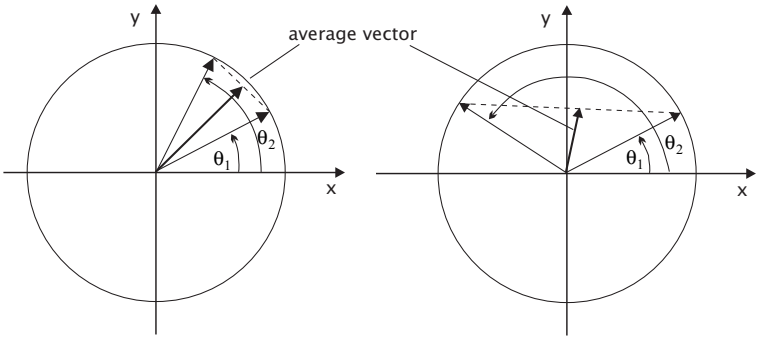

Figure 11.20: *Averaging of a cyclic quantity represented as a normal vector* $\bar{\boldsymbol{n}}_\theta = [\cos\theta, \sin\theta]^T$ *on the unit vector. The average vector* $(\bar{\boldsymbol{n}}_{\theta_1} + \bar{\boldsymbol{n}}_{\theta_2})/2$ *points in the correct direction* $(\theta_1+\theta_2)/2$ *but its magnitude decreases with the difference angle.*

Finally, we consider a measure to characterize the scatter in the direction of vectors. Figure 11.20 illustrates that for low scatter the sum vector is only slightly lower than the sum of the magnitudes of the vector. Thus, we can define an angular coherence measure as

$$c = \frac{|\boldsymbol{H} * \mathbf{G}|}{|\mathbf{G}|}, \tag{11.58}$$

where $\boldsymbol{H}$ is an arbitrary smoothing convolution operator. This measure is one if all vectors in the neighborhood covered by the convolution operator point in the same direction and zero if they are equally distributed. This definition of a coherence measure works not only in two-dimensional but also in higher-dimensional vector spaces. In one-dimensional vector spaces (scalar images), the coherency measure is, of course, always one.

11.9 Further Readings‡

The articles of Simonds [171] and Wells [197] discuss fast algorithms for large Gaussian kernels. Readers with an interest in the general principles of efficient algorithms are referred to the textbooks of Aho et al. [3] or Sedgewick [167]. Blahut [10] deals with fast algorithms for digital signal processing. Classical filter design techniques, especially for IIR-filter are discussed in the standard textbooks for signal processing, e. g., Proakis and Manolakis [144] or Oppenheim and Schafer [133]. Lim [112] specifically deals with the design of 2-D IIR filters. A detailed description of the deterministic and statistical properties of *median filters* can be found in Huang [75] or Arce et al. [5]. They are also discussed in detail in the monograph on nonlinear digital filters by Pitas and Venetsanopoulos [140]. The monograph of Granlund and Knutsson [57] on signal processing for computer vision deals also with weighted averaging (normalized convolution, Section 11.7.2). Steerable filters (Section 11.7.3) were introduced by the articles of Freeman and Adelson [48] and Simoncelli et al. [170].

12 Edges

12.1 Introduction

The task of *edge detection* requires neighborhood operators that are sensitive to changes and suppress areas of constant gray values. In this way, a feature image is formed in which those parts of the image appear bright where changes occur while all other parts remain dark.

Mathematically speaking, an ideal edge is a discontinuity of the spatial gray value function $g(\boldsymbol{x})$ of the image plane. It is obvious that this is only an abstraction, which often does not match the reality. Thus, the first task of edge detection is to find out the properties of the edges contained in the image to be analyzed. Only if we can formulate a model of the edges, can we determine how accurately and under what conditions it will be possible to detect an edge and to optimize edge detection.

Edge detection is always based on *differentiation* in one or the other form. In discrete images, differentiation is replaced by *discrete differences*, which only approximate to differentiation. The errors associated with these approximations require careful consideration. They cause effects that are not expected in the first place. The two most serious errors are: anisotropic edge detection, i. e., edges are not detected equally well in all directions, and erroneous estimation of the direction of the edges.

While the definition of edges is obvious in scalar images, different definitions are possible in multicomponent or vectorial images (Section 12.6). An edge might be a feature that shows up in only one component or in all. Edge detection also becomes more complex in higher-dimensional images. In three dimensions, for example, volumetric regions are separated by surfaces, and edges become discontinuities in the orientation of surfaces.

Another important question is the reliability of the edge estimates. We do not only want to find an edge but also to know how significant it is. Thus, we need a measure for *edge strength*. Closely related to this issue is the question of optimum edge detection. Once edge detectors deliver not only edges but also an objective confidence measure, different edge detectors can be compared to each other and optimization of edge detection becomes possible.

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2

Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

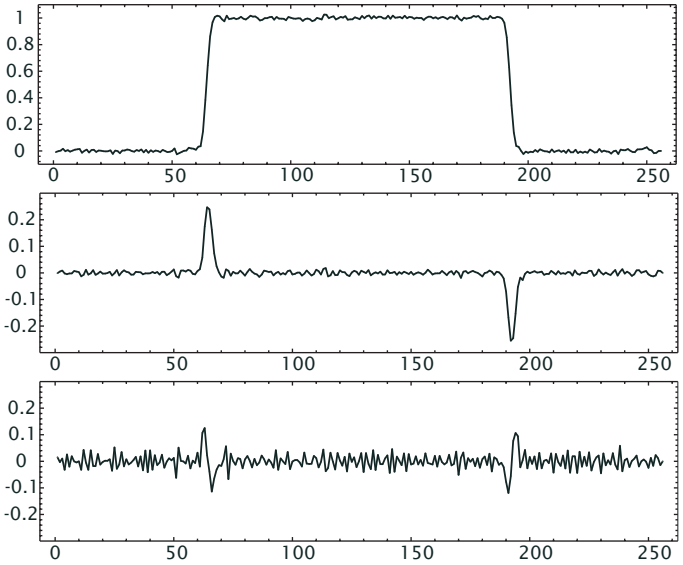

Figure 12.1: *Noisy 1-D edge and its first and second derivative.*

12.2 General Properties of Edge Filters

Averaging filters suppress structures with high wave numbers. Edge detection requires a filter operation that emphasizes the changes in gray values and suppresses areas with constant gray values. Figure 12.1 illustrates that derivative operators are suitable for such an operation. The first derivative shows an extreme at the edge, while the second derivative crosses zero where the edge has its steepest ascent or descent. Both criteria can be used to detect edges.

A pth-order *partial derivative* operator corresponds to multiplication by $(\mathrm{i}k)^p$ in the wave number space (Section 2.3, ≻ R4):

$$\frac{\partial}{\partial x_w} \quad \circ\!\!-\!\!\bullet \quad 2\pi \mathrm{i} k_w, \qquad \frac{\partial^2}{\partial x_w^2} \quad \circ\!\!-\!\!\bullet \quad -4\pi^2 k_w^2. \tag{12.1}$$

The first-order partial derivatives into all directions of a W-dimensional signal $g(\boldsymbol{x})$ form the W-dimensional *gradient vector*:

$$\nabla g(\boldsymbol{x}) = \left[\frac{\partial g}{\partial x_1}, \frac{\partial g}{\partial x_2}, \ldots, \frac{\partial g}{\partial x_W}\right]^T. \tag{12.2}$$

The magnitude of the gradient vector,

$$|\nabla g| = ||\nabla g||_2 = \left(\nabla g^T \nabla g\right)^{1/2} = \left(\sum_{w=1}^{W} \left(\frac{\partial g}{\partial x_w}\right)^2\right)^{1/2}, \tag{12.3}$$

is invariant on rotation of the coordinate system.

All possible combinations of second-order partial derivates of a W-dimensional signal form a symmetric $W \times W$ matrix, known as the *Hesse matrix*:

$$\boldsymbol{H} = \begin{bmatrix} \frac{\partial g^2}{\partial x_1^2} & \frac{\partial g^2}{\partial x_1 x_2} & \cdots & \frac{\partial g^2}{\partial x_1 x_W} \\ \frac{\partial g^2}{\partial x_1 x_2} & \frac{\partial g^2}{\partial x_2^2} & \cdots & \frac{\partial g^2}{\partial x_2 x_W} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g^2}{\partial x_1 x_W} & \frac{\partial g^2}{\partial x_2 x_W} & \cdots & \frac{\partial g^2}{\partial x_W^2} \end{bmatrix}. \tag{12.4}$$

The trace of this matrix, i. e., the sum of the diagonal is called the *Laplacian operator* and is denoted by Δ:

$$\Delta = \text{trace}\,\boldsymbol{H} = \sum_{w=1}^{W} \frac{\partial^2}{\partial x_w^2} \quad \circ\!\!-\!\!\bullet \quad -\sum_{w=1}^{W} k_w^2 = -k^2. \tag{12.5}$$

As the magnitude of the gradient, the Laplace operator is invariant upon rotation of the coordinate system.

In Sections 12.2.1–12.2.4, we discuss the general properties of filters that form the basis of edge detection. This discussion is similar to that on the general properties of averaging filters in Sections 11.2.1–11.2.4.

12.2.1 Zero Shift

With respect to object detection, the most important feature of a derivative convolution operator is that it must not shift the object position. For a smoothing filter, this constraint required a real transfer function and a symmetric convolution mask (Section 11.2.1). For a first-order derivative filter, a real transfer function makes no sense, as extreme values should be mapped onto zero crossings and the steepest slopes to extreme values. This mapping implies a 90° phase shift. Therefore, the transfer function of a first-order derivative filter must be imaginary. An imaginary transfer function implies an antisymmetric filter mask. An antisymmetric convolution mask is defined as

$$h_{-n} = -h_n. \tag{12.6}$$

For a convolution mask with an odd number of coefficients, this implies that the central coefficient is zero.

A second-order derivative filter detects curvature. Extremes in function values should coincide with extremes in curvature. Consequently, a second-order derivative filter should be symmetric, like a smoothing

filter. All the symmetric filter properties discussed for smoothing filters also apply to these filters (Section 11.2.1).

12.2.2 Suppression of Mean Value

A derivative filter of any order must not show response to constant values or an offset in a signal. This condition implies that the sum of the coefficients must be zero and that the transfer function is zero for a zero wave number:

$$\begin{aligned} &\text{1-D:} \quad \hat{h}(0) = 0, \quad \sum_n h_n = 0 \\ &\text{2-D:} \quad \hat{h}(\mathbf{0}) = 0, \quad \sum_m \sum_n h_{mn} = 0 \\ &\text{3-D:} \quad \hat{h}(\mathbf{0}) = 0, \quad \sum_l \sum_m \sum_n h_{lmn} = 0. \end{aligned} \tag{12.7}$$

Also, a second-order derivative filter should not respond to a constant slope. This condition implies no further constraints as it can be derived from the symmetry of the filter and the zero sum condition Eq. (12.7).

12.2.3 Symmetry Properties

The symmetry properties deserve further consideration as they form the basis for computing the convolution more efficiently by reducing the number of multiplications and simplifying the computations of the transfer functions. The zero-shift condition (Section 12.2.1) implies that a first-order derivative filter generally has a 1-D mask of odd symmetry with $2R + 1$ or $2R$ coefficients:

$$[h_1, \ldots, h_1, 0, -h_1, \ldots, -h_R] \quad \text{or} \quad [h_{R1}, \ldots, h_1, -h_1, \ldots, -h_R]. \tag{12.8}$$

Therefore, the computation of the convolution reduces to

$$g'_n = \sum_{n'=1}^{R} h_{n'} (g_{n-n'} - g_{n+n'}) \quad \text{or} \quad g'_{n+1/2} = \sum_{n'=1}^{R} h_{n'} (g_{n+1-n'} - g_{n+n'}). \tag{12.9}$$

For $2R + 1$ filter coefficients only R multiplications are required. The number of additions, however, is still $2R - 1$.

The symmetry relations also significantly ease the computation of the transfer functions because only the sine terms of the complex exponential from the Fourier transform remain in the equations. The transfer functions for a 1-D odd mask is

$$\hat{g}(\tilde{k}) = 2\mathrm{i} \sum_{v=1}^{R} h_v \sin(v\pi\tilde{k}) \quad \text{or} \quad \hat{g}(\tilde{k}) = 2\mathrm{i} \sum_{v=1}^{R} h_v \sin[(v - 1/2)\pi\tilde{k}]. \tag{12.10}$$

For second-order derivative filters, we can use all the equations derived for the averaging filters in Section 11.2.1, as these feature an even symmetry in the direction of the deviation.

12.2.4 Nonselective Derivation and Isotropy

Intuitively, we expect that any derivative operator amplifies smaller scales more strongly than coarser scales, because the transfer function of an ideal derivative operator goes with k^w (Eq. (12.1)). Consequently, we could argue that the transfer function of a good discrete derivative operator should approximate the ideal transfer functions in Eq. (12.1) as close as possible.

However, this condition is a too strong restriction. The reason is the following. Imagine that we first apply a smoothing operator to an image before we apply a derivative operator. We would still recognize the joint operation as a derivation. The mean gray value is suppressed and the operator is still only sensitive to spatial gray value changes.

Therefore, the ideal transfer function in Eq. (12.1) could be restricted to small wave numbers:

$$\hat{h}(\boldsymbol{k})\Big|_{k_w=0} = 0, \quad \frac{\partial \hat{h}(\boldsymbol{k})}{\partial k_w}\Big|_{k_w=0} = (\mathrm{i}k_w)^p, \quad \frac{\partial^2 \hat{h}(\boldsymbol{k})}{\partial k_w^2}\Big|_{k_w=0} = 0. \tag{12.11}$$

For good edge detection, it is important that the response of the operator does not depend on the direction of the edge. If this is the case, we speak of an *isotropic edge detector*. The isotropy of an edge detector can best be analyzed by its transfer function. The most general form for an isotropic derivative operator of order p is given by

$$\hat{h}(\boldsymbol{k}) = (\mathrm{i}k_w)^p \hat{b}(|\tilde{\boldsymbol{k}}|) \quad \text{with} \quad \hat{b}(0) = 1 \quad \text{and} \quad \nabla_k \hat{b}(\boldsymbol{k}) = \boldsymbol{0}. \tag{12.12}$$

The constraints for derivative operators are summarized in Appendix A (≻ R24 and ≻ R25).

12.3 Gradient-Based Edge Detection†

12.3.1 Principle

In terms of first-order changes, an edge is defined as an extreme (Fig. 12.1). Thus edge detection with first-order derivative operators means to search for the steepest changes, i. e., maxima of the magnitude of the gradient vector (Eq. (12.2)). Therefore, first-order partial derivatives in all directions must be computed. In the operator notation, the gradient can be

written as a vector operator. In 2-D and 3-D space this is

$$\mathcal{D} = \begin{bmatrix} \mathcal{D}_x \\ \mathcal{D}_y \end{bmatrix} \quad \text{or} \quad \mathcal{D} = \begin{bmatrix} \mathcal{D}_x \\ \mathcal{D}_y \\ \mathcal{D}_z \end{bmatrix}. \tag{12.13}$$

Because the gradient is a vector, its magnitude (Eq. (12.3)) is invariant upon rotation of the coordinate system. This is a necessary condition for isotropic edge detection. The computation of the magnitude of the gradient can be expressed in the 2-D space by the operator equation

$$|\mathcal{D}| = \left[\mathcal{D}_x \cdot \mathcal{D}_x + \mathcal{D}_y \cdot \mathcal{D}_y\right]^{1/2}. \tag{12.14}$$

The symbol $\cdot$ denotes a pointwise multiplication of the images that result from the filtering with the operators $\mathcal{D}_x$ and $\mathcal{D}_y$, respectively (Section 4.1.4). Likewise, the square root is performed pointwise in the space domain. According to Eq. (12.14), the application of the operator $|\mathcal{D}|$ to the image $\boldsymbol{G}$ means the following chain of operations:

1. filter the image $\boldsymbol{G}$ independently with $\mathcal{D}_x$ and $\mathcal{D}_y$,
2. square the gray values of the two resulting images,
3. add the resulting images, and
4. compute the square root of the sum.

At first glance it appears that the computation of the magnitude of the gradient is computationally expensive. Therefore it is often approximated by

$$|\mathcal{D}| \approx |\mathcal{D}_x| + \left|\mathcal{D}_y\right|. \tag{12.15}$$

However, this approximation is anisotropic even for small wave numbers. It detects edges along the diagonals $\sqrt{2}$ times more sensitively than along the principal axes. The computation of the magnitude of the gradient can, however, be performed as a *dyadic point operator* efficiently by a *look-up table* (Section 10.4.2).

12.3.2 Error in magnitude and direction

The principal problem with all types of edge detectors is that on a discrete grid a derivative operator can only be approximated. In general, two types of errors result from this approximation (Fig. 12.2). First, edge detection will become anisotropic, i. e., the computation of the magnitude of the gradient operator depends on the direction of the edge. Second, the direction of the edge deviates from the correct direction. For both types of errors it is useful to introduce error measures. All error measures are computed from the transfer functions of the gradient filter operator.

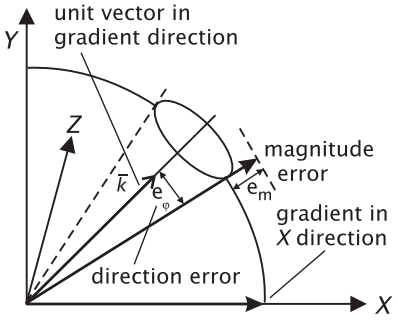

Figure 12.2: *Illustration of the magnitude and direction error of the gradient vector.*

The magnitude of the gradient is then given by

$$\left|\hat{\boldsymbol{d}}(\boldsymbol{k})\right| = \left(\hat{d}_x(\boldsymbol{k})^2 + \hat{d}_y(\boldsymbol{k})^2\right)^{1/2}, \tag{12.16}$$

where $\hat{\boldsymbol{d}}(\boldsymbol{k})$ is the vectorial transfer function of the gradient operator. The anisotropy in the magnitude of the gradient can then be expressed by the deviation of the magnitude from the magnitude of the gradient in x direction, which is given by

$$e_m(\boldsymbol{k}) = \left|\hat{\boldsymbol{d}}(\boldsymbol{k})\right| - \left|\hat{d}_x(\boldsymbol{k})\right|. \tag{12.17}$$

This error measure can be used for signals of any dimension.

In a similar way, the error in the direction of the gradient can be computed. From the components of the gradient, the computed angle ϕ' of the 2-D gradient vector is

$$\phi' = \arctan \frac{\hat{d}_y(k,\phi)}{\hat{d}_x(k,\phi)}. \tag{12.18}$$

The error in the angle is therefore given by

$$e_\phi(k,\phi) = \arctan \frac{\hat{d}_y(k,\phi)}{\hat{d}_x(k,\phi)} - \phi. \tag{12.19}$$

In higher dimensions, angle derivation can be in different directions. Still we can find a direction error by using the scalar product between a unit vector in the direction of the true gradient vector and the computed gradient vector $\hat{\boldsymbol{d}}(k)$ (Fig. 12.2):

$$\cos e_\varphi = \frac{\bar{\boldsymbol{k}}^T \hat{\boldsymbol{d}}(\boldsymbol{k})}{\left|\hat{\boldsymbol{d}}(\boldsymbol{k})\right|} \quad \text{with} \quad \bar{\boldsymbol{k}} = \frac{\boldsymbol{k}}{|\boldsymbol{k}|}. \tag{12.20}$$

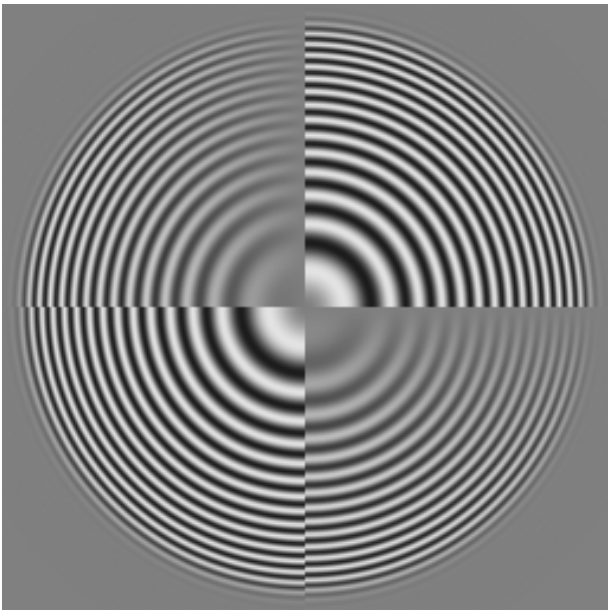

Figure 12.3: *Application of the first-order symmetric derivative filters $\mathcal{D}_x$ and $\mathcal{D}_y$ to the test image shown in Fig. 11.4.*

In contrast to the angle error measure (Eq. (12.19)) for two dimensions, this error measure has only positive values. It is a scalar and thus cannot give the direction of the deviation.

A wide variety of solutions for edge detectors exist. We will discuss some of them carefully in Sections 12.3.3–12.3.5.

12.3.3 First-Order Discrete Differences

First-order discrete differences are the simplest of all approaches to compute a gradient vector. For the first partial derivative in the x direction, one of the following approximations for $\partial g(x_1, x_2)/\partial x_1$ may be used:

$$\begin{array}{ll} \text{Backward difference} & \dfrac{g(x_1, x_2) - g(x_1 - \Delta x_1, x_2)}{\Delta x_1} \\ \text{Forward difference} & \dfrac{g(x_1 + \Delta x_1, x_2) - g(x_1, x_2)}{\Delta x_1} \\ \text{Symmetric difference} & \dfrac{g(x_1 + \Delta x_1, x_2) - g(x_1 - \Delta x_1, x_2)}{2\Delta x_1}. \end{array} \tag{12.21}$$

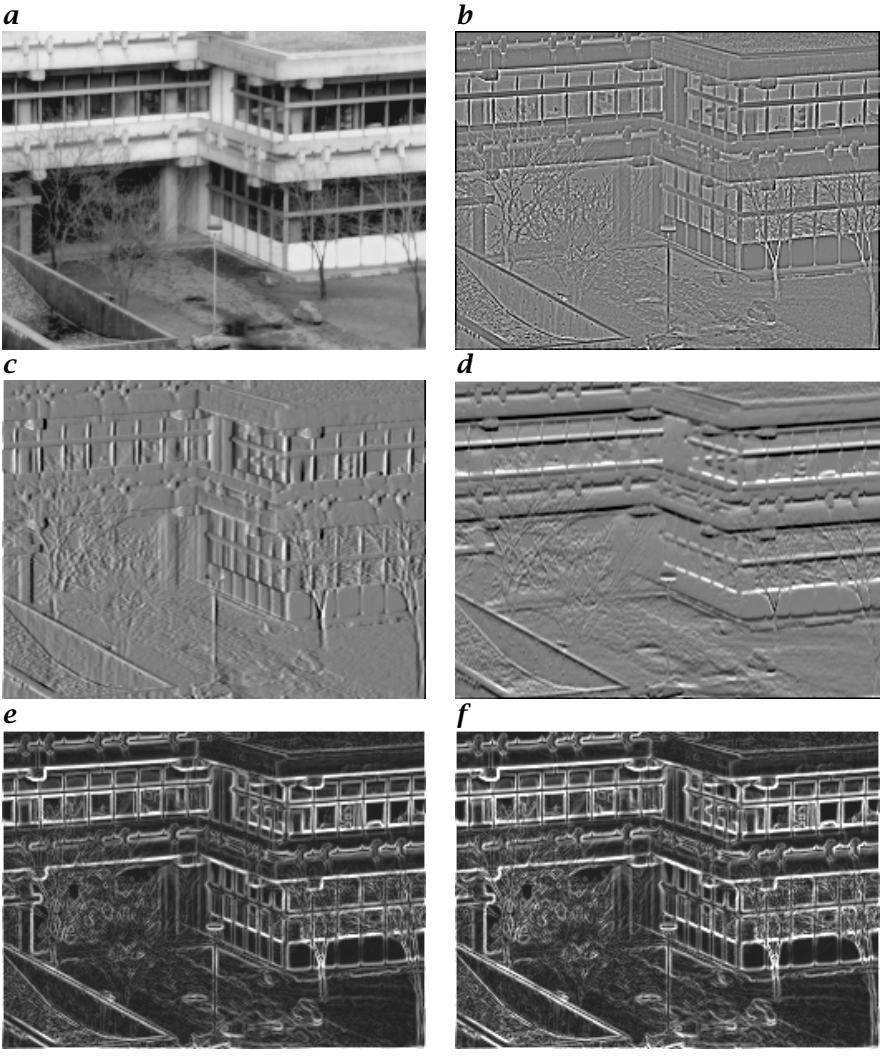

Figure 12.4: *Detection of edges by derivative filters:* ***a*** *Original image,* ***b*** *Laplacian operator* $\mathcal{L}$*,* ***c*** *horizontal derivative* $\mathcal{D}_{2x}$*,* ***d*** *vertical derivative* $\mathcal{D}_{2y}$*,* ***e*** *magnitude of the gradient* $(\mathcal{D}_{2x} \cdot \mathcal{D}_{2x} + \mathcal{D}_{2y} \cdot \mathcal{D}_{2y})^{1/2}$*, and* ***f*** *sum of the magnitudes of* ***c*** *and* ***d*** *after Eq. (12.15).*

These approximations correspond to the filter masks

$$\begin{array}{lll} \text{Backward} & {}^{-}\boldsymbol{D}_x & = [1\boldsymbol{.}\ -1] \\ \text{Forward} & {}^{+}\boldsymbol{D}_x & = [1\ -1\boldsymbol{.}] \\ \text{Symmetric} & \boldsymbol{D}_{2x} & = 1/2\ [1\ 0\ -1]\,. \end{array} \tag{12.22}$$

The subscript • denotes the central pixel of the asymmetric masks with two elements. Only the last mask shows the symmetry properties required in Section 12.2.3. We may also consider the two-element masks corresponding to the backward or forward difference as odd masks provided that the result is not stored at the position of the right or left pixel but at a position halfway between the two pixels. This corresponds to a shift of the grid by half a pixel distance. The transfer function for the backward difference is then

$$^{-}\hat{d}_x = \exp(\mathrm{i}\pi\tilde{k}_x/2)\left[1 - \exp(-\mathrm{i}\pi\tilde{k}_x)\right] = 2\mathrm{i}\sin(\pi\tilde{k}_x/2), \tag{12.23}$$

where the first term results from the shift by half a grid point.

Using Eq. (12.10), the transfer function of the symmetric difference operator reduces to

$$\hat{d}_{2x} = \mathrm{i}\sin(\pi\tilde{k}_x) = \mathrm{i}\sin(\pi\tilde{k}\cos\phi). \tag{12.24}$$

This operator can also be computed from

$$\boldsymbol{D}_{2x} = {}^{-}\boldsymbol{D}_x\,{}^{1}\boldsymbol{B}_x = [1_\bullet\ -1] * 1/2\ [1\ 1_\bullet] = 1/2\,[1\ 0\ -1].$$

The first-order difference filters in other directions are given by similar equations. The transfer function of the symmetric difference filter in y direction is, e. g., given by

$$\hat{d}_{2y} = \mathrm{i}\sin(\pi\tilde{k}_y) = \mathrm{i}\sin(\pi\tilde{k}\sin\phi). \tag{12.25}$$

The application of $\mathcal{D}_{2x}$ to the ring test pattern in Fig. 12.3 illustrates the directional properties and the 90° phase shift of these filters. Figure 12.4 shows the detection of edges with these filters, the magnitude of the gradient, and the sum of the magnitudes of $\mathcal{D}_{2x}$ and $\mathcal{D}_{2y}$.

Unfortunately, these simple difference filters are only poor approximations for an edge detector. From Eqs. (12.24) and (12.25), we infer that the magnitude and direction of the gradient are given by

$$|\hat{\boldsymbol{d}}| = \left(\sin^2(\pi\tilde{k}\cos\phi) + \sin^2(\pi\tilde{k}\sin\phi)\right)^{1/2} \tag{12.26}$$

and

$$\phi' = \arctan\frac{\sin^2(\pi\tilde{k}\sin\phi)}{\sin(\pi\tilde{k}\cos\phi)}, \tag{12.27}$$

where the wave number is written in polar coordinates (k, ϕ). The resulting errors are shown in a pseudo 3-D plot in Fig. 12.5 as a function of the magnitude of the wave number and the angle to the x axis. The magnitude of the gradient decreases quickly from the correct value. A Taylor expansion of Eq. (12.26) in $\tilde{k}$ yields for the relative error in the magnitude

$$e_m(\tilde{k}, \phi) \approx \frac{(\pi\tilde{k})^3}{12}\sin^2 2\phi + O(\tilde{k}^5). \tag{12.28}$$

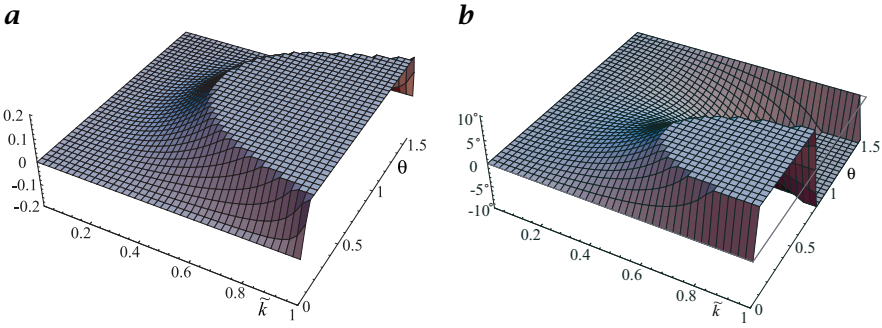

Figure 12.5: ***a*** *Anisotropy of the magnitude and* ***b*** *error in the direction of the gradient based on the symmetrical gradient operator* $\left[\mathcal{D}_{2x}, \mathcal{D}_{2y}\right]^T$. *The parameters are the magnitude of the wave number (0 to 1) and the angle to the x axis (0 to* $\pi/2$*).*

The decrease is also anisotropic; it is slower in the diagonal direction. The errors in the direction of the gradient are also large (Fig. 12.5b). While in the direction of the axes and diagonals the error is zero, in the directions in between it reaches values of about ± 10° at $\tilde{k} = 0.5$. A Taylor expansion of Eq. (12.27) in $\tilde{k}$ yields the angle error according to Eq. (12.19) in the approximation for small $\tilde{k}$:

$$e_\phi(\tilde{k}, \phi) \approx \frac{(\pi\tilde{k})^2}{24} \sin 4\phi + O(\tilde{k}^4). \tag{12.29}$$

As observed in Fig. 12.5b, the angle error is zero for $\phi = n\pi/4$ with $n \in \mathbb{Z}$, i. e., for $\phi = 0°, 45°\ 90°, \ldots$

12.3.4 Spline-Based Edge Detection‡

The cubic B-spline transform discussed in Section 10.6.5 for interpolation yields a continuous representation of a discrete image that is also continuous in its first and second derivative:

$$g_3(x) = \sum_n c_n \beta_3(x - n), \tag{12.30}$$

where $\beta_3(x)$ is the cubic B-spline function defined in Eq. (10.60). From this continuous representation, it is easy to compute the spatial derivative of $g_3(x)$:

$$\frac{\partial g_3(x)}{\partial x} = \sum_n c_n \frac{\partial \beta_3(x - n)}{\partial x}. \tag{12.31}$$

For a discrete derivative filter, we only need the derivatives at the grid points. From Fig. 10.20a it can be seen that the cubic B-spline function covers at most 5 grid points. The maximum of the spline function occurs at the central grid point. Therefore, the derivative at this point is zero. It is also zero at the two

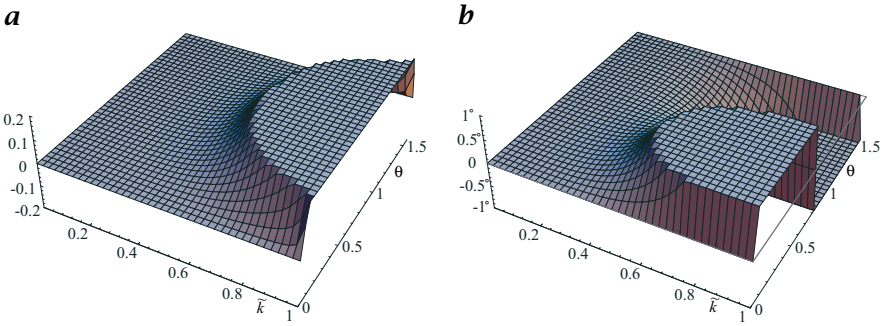

Figure 12.6: ***a*** *Anisotropy of the magnitude and* ***b*** *error in the direction of the gradient based on the cubic B-spline derivative operator according to Eq. (12.33). Parameters are the magnitude of the wave number (0 to 1) and the angle to the x axis (0 to $\pi/2$).*

outer grid points. Thus, the derivative is only unequal to zero at the direct left and right neighbors of the central point. Therefore, the derivative at the grid point x_m reduces to

$$\left.\frac{\partial g_3(x)}{\partial x}\right|_{x_m} = (c_{m+1} - c_{m-1})/2. \tag{12.32}$$

Thus the computation of the first-order derivative based on the cubic B-spline transformation is indeed an efficient solution. We apply first the cubic B-spline transform in the direction of the derivative to be computed (Section 10.6.5) and then the $\mathcal{D}_{2x}$ operator. Therefore, the transfer function is given by

$$\hat{\boldsymbol{D}}_x = \mathrm{i}\frac{\sin(\pi\tilde{k}_x)}{2/3 + 1/3\cos(\pi\tilde{k}_x)} = \mathrm{i}\pi\tilde{k}_x - \mathrm{i}\frac{\pi^5\tilde{k}_x^5}{180} + O(\tilde{k}_x^7). \tag{12.33}$$

The errors in the magnitude and direction of a gradient vector based on the B-spline derivative filter are shown in Fig. 12.6. They are considerably less than for the simple difference filters (Fig. 12.5). This can be seen more quantitatively from Taylor expansions for the relative errors in the magnitude of the gradient

$$e_m(\tilde{k},\phi) \approx -\frac{(\pi\tilde{k})^5}{240}\sin^2 2\phi + O(\tilde{k}^7) \tag{12.34}$$

and the angle error

$$e_\phi(\tilde{k},\phi) \approx \frac{(\pi\tilde{k})^4}{720}\sin 4\phi + O(\tilde{k}^6). \tag{12.35}$$

The error terms are now contained only in terms with $\tilde{k}^4$ (and higher powers of $\tilde{k}$). Compare also Eqs. (12.34) and (12.35) with Eqs. (12.28) and (12.29).

12.3.5 Least-Squares Optimized Gradient‡

In this section first-order derivative filters are discussed that have been optimized using the least squares technique already used in Section 10.6.6 to optimize interpolation filters. The basic idea is to use a one-dimensional $2R+1$

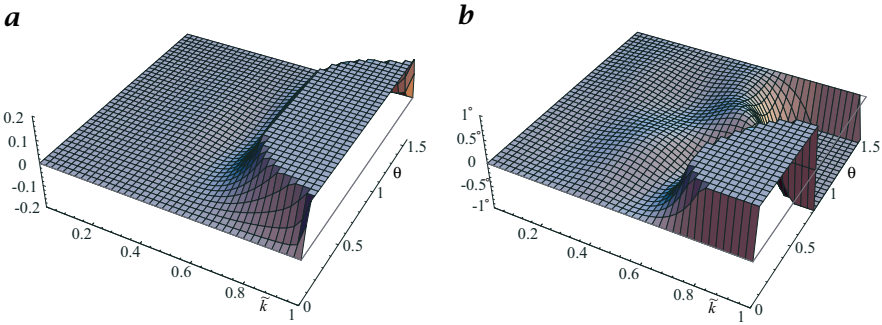

Figure 12.7: ***a*** *Anisotropy of the magnitude and* ***b*** *error in the direction of the gradient based on the least squares optimized derivative filter according to Eq. (12.39) for* $R = 3$ *(*$d_1 = -0.597949, d_2 = 0.189835$, $d_3 = -0.0357216$*). Parameters are the magnitude of the wave number (0 to 1) and the angle to the* x *axis (0 to* $\pi/2$*).*

filter mask with odd symmetry in the corresponding direction w and to vary the coefficients so that the transfer function approximates the ideal transfer function of a derivative filter, $\mathrm{i}\pi\tilde{k}_w$, with a minimum deviation. Thus the *target function* is

$$\hat{t}(\tilde{k}_w) = \mathrm{i}\pi\tilde{k}_w \tag{12.36}$$

and the transfer function of a one-dimensional $2R+1$ filter with R unknown coefficients is

$$^R\hat{d}(\tilde{k}_w) = -\mathrm{i}\sum_{v=1}^{R} 2d_v \sin(v\pi\tilde{k}_w). \tag{12.37}$$

As for the interpolation filters in Section 10.6.6, the coefficients are determined in such a way that $^R\hat{d}(\tilde{\boldsymbol{k}})$ shows a minimum deviation from $\hat{t}(\tilde{\boldsymbol{k}})$ in the least-squares sense:

$$\int_0^1 w(\tilde{k}_w)\left|{}^R\hat{d}(\tilde{k}_w) - \hat{t}(\tilde{k}_w)\right|^2 \mathrm{d}\tilde{k}_w. \tag{12.38}$$

The wave number-dependent weighting function $w(\tilde{k}_w)$ determines the weighting of the individual wave numbers.

One useful additional constraint is to force the transfer function to be equal to $\mathrm{i}\pi\tilde{k}$ for small wave numbers. This constraint reduces the degree of freedom by one for a filter with R coefficients, so only $R-1$ can be varied. The resulting equations are

$$^R\hat{d} = -\mathrm{i}\sin(\pi\tilde{k}_w) - \mathrm{i}\sum_{v=2}^{R} 2d_v\left(\sin(v\pi\tilde{k}_w) - v\sin(\pi\tilde{k}_w)\right) \tag{12.39}$$

and

$$d_1 = 1 - \sum_{v=2}^{R} v d_v. \tag{12.40}$$

As a comparison of Figs. 12.6 and 12.7 shows, this filter exhibits a significantly lower error than a filter designed with the cubic B-spline interpolation.

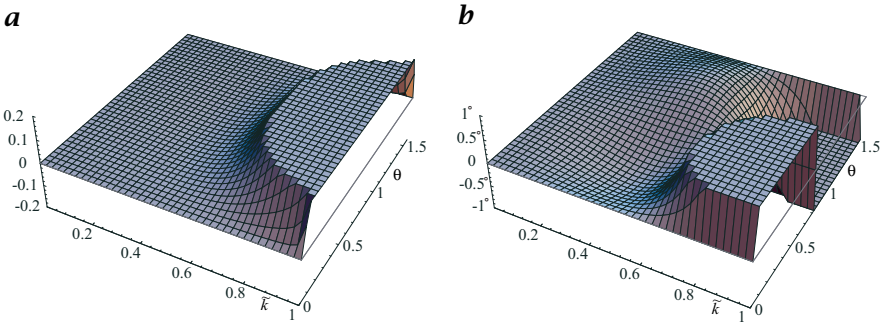

Figure 12.8: ***a*** *Anisotropy of the magnitude and* ***b*** *error in the direction of the gradient based on the least squares recursive derivative filter according to Eq. (12.41) for $R = 2$ ($\beta = -0.439496$, $d_1 = -0.440850$, $d_2 = -0.0305482$. Parameters are the magnitude of the wave number (0 to 1) and the angle to the x axis (0 to $\pi/2$).*

Derivative filters can be further improved by compensating the decrease in the transfer function by a forward and backward running recursive relaxation filter (Section 4.3.5, Fig. 4.4b). Then the resulting transfer function is

$$ {}^{(R,\beta)}\hat{d} = \frac{-\mathrm{i}\sin(\pi\tilde{k}) - \mathrm{i}\sum_{v=2}^{R} 2d_v \left(\sin(v\pi\tilde{k}_w) - v\sin(\pi\tilde{k}_w)\right)}{1+\beta-\beta\cos(\pi\tilde{k}_w)} \tag{12.41} $$

with the additional parameter β. Figure 12.8 shows the errors in the magnitude and direction of the gradient for $R = 2$.

A more detailed discussion on the design of optimal derivative filters including tables with filter coefficients can be found in Jähne [81].

12.4 Edge Detection by Zero Crossings

12.4.1 Principle

Edges constitute *zero crossings* in second-order derivatives (Fig. 12.1). Therefore, the second-order derivatives in all directions can simply be added up to form a linear isotropic edge detector with the transfer function $-(\pi\tilde{k})^2$ (Eq. (12.5)), known as the *Laplacian operator* . From Fig. 12.1 it is also obvious that not every zero crossing constitutes an edge.

Only peaks before and after a zero that are significantly higher than the noise level indicate valid edges. From Fig. 12.1 we can also conclude that edge detection with the Laplace operator is obviously much more sensitive to noise in the signal than edge detection using a gradient-based approach.

12.4.2 Laplace Filter

We can directly derive second-order derivative operators by a twofold application of first-order operators

$$\mathcal{D}_x^2 = {}^-\mathcal{D}_x\,{}^+\mathcal{D}_x. \tag{12.42}$$

In the spatial domain, this means

$$[1\bullet\ -1] * [1\ -1\bullet] = [1\ -2\ 1]. \tag{12.43}$$

The discrete Laplace operator $\mathcal{L} = \mathcal{D}_x^2 + \mathcal{D}_y^2$ for 2-D images thus has the filter mask

$$\boldsymbol{L} = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} + \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{12.44}$$

and the transfer function

$$\hat{l}(\tilde{\boldsymbol{k}}) = -4\sin^2(\pi\tilde{k}_x/2) - 4\sin^2(\pi\tilde{k}_y/2). \tag{12.45}$$

Like other discrete approximations of operators, the Laplace operator is only isotropic for small wave numbers (Fig. 12.9a):

$$\hat{l}(\tilde{k},\phi) = -(\pi\tilde{k})^2 + \frac{3}{48}(\pi\tilde{k})^4 + \frac{1}{48}\cos 4\phi(\pi\tilde{k})^4 + O(\tilde{k}^6). \tag{12.46}$$

There are many other ways to construct a discrete approximation for the Laplace operator. An interesting possibility is the use of binomial masks. With Eq. (11.23) we can approximate all binomial masks for sufficiently small wave numbers by

$$\hat{b}^{2R}(\tilde{k}) \approx 1 - \frac{R}{4}(\tilde{k}\pi)^2 + O(\tilde{k}^4). \tag{12.47}$$

From this equation we can conclude that any operator $\mathcal{B}^p - \mathcal{I}$ constitutes a Laplace operator for small wave numbers. For example,

$$\begin{aligned} \boldsymbol{L}' = 4(\boldsymbol{B}^2 - \boldsymbol{I}) &= \left[\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right] \\ &= \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix} \end{aligned} \tag{12.48}$$

with the transfer function

$$\hat{l}'(\tilde{\boldsymbol{k}}) = 4\cos^2(\pi\tilde{k}_x/2)\cos^2(\pi\tilde{k}_y/2) - 4 \tag{12.49}$$

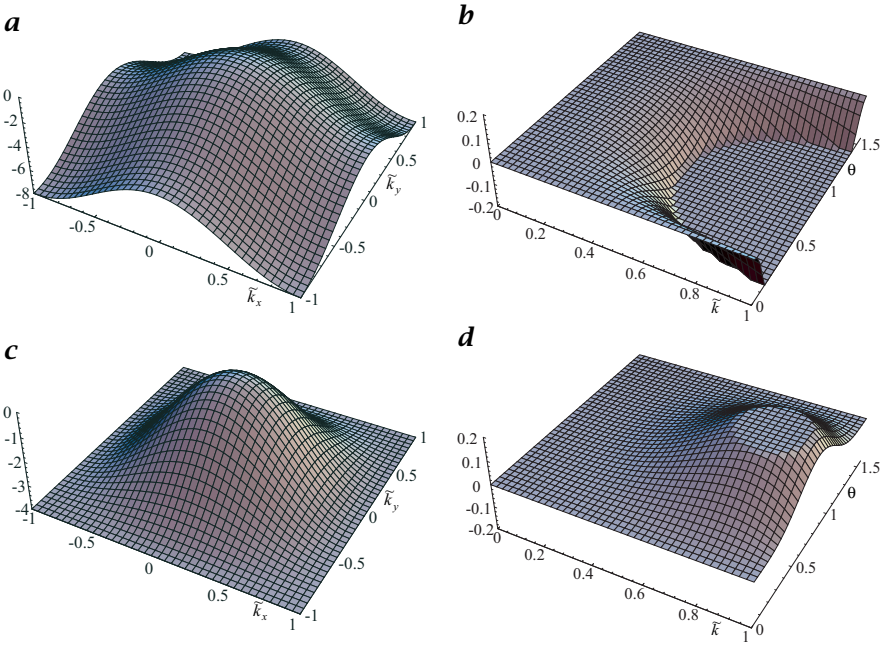

Figure 12.9: *Transfer functions of discrete Laplace operators and their anisotropy:* **a** $\mathcal{L}$ *Eq. (12.44),* **b** $\hat{l}(k,\theta)-\hat{l}(k,0)$, **c** $\mathcal{L}'$ *Eq. (12.48),* **d** $\hat{l}'(k,\theta)-\hat{l}'(k,0)$.

is another example of a discrete Laplacian operator. For small wave numbers it can be approximated by

$$\hat{l}'(\tilde{k},\phi) \approx -(\pi\tilde{k})^2 + \frac{3}{32}(\pi\tilde{k})^4 - \frac{1}{96}\cos 4\phi(\pi\tilde{k})^4 + O(\tilde{k}^6). \qquad (12.50)$$

For large wave numbers, the transfer functions of both Laplace operators show considerable deviations from an ideal Laplacian, $-(\pi\tilde{k})^2$. $\mathcal{L}'$ is significantly less anisotropic than $\mathcal{L}$ (Fig. 12.9).

12.5 Regularized Edge Detection

12.5.1 Principle

The edge detectors discussed so far are still poor performers, especially in noisy images. Because of their small mask sizes, they are most sensitive to high wave numbers. At high wave numbers there is often more noise than signal in images. In other words, we have not yet considered the importance of scales for image processing as discussed in Section 5.1.1. Thus, the way to optimum edge detectors lies in the tuning of edge detectors to the scale (wave number range) with the maximum

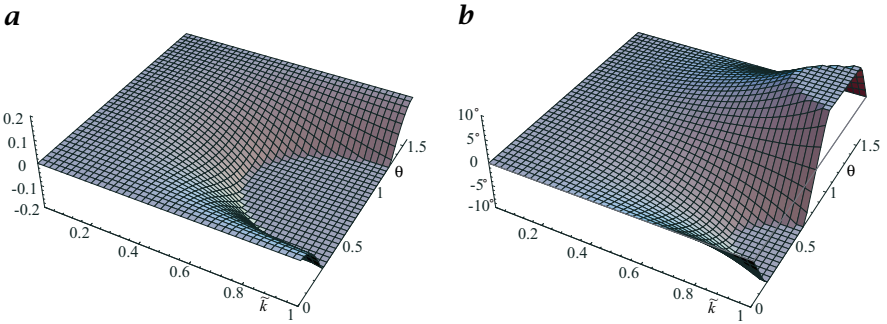

Figure 12.10: ***a*** *Anisotropy of the magnitude and* ***b*** *error in the direction of the gradient based on the* 2×2 *cross-smoothing edge detector Eq. (12.51). The parameters are the magnitude of the wave number (0 to 1) and the angle to the* x *axis (0 to* $\pi/2$*).*

signal-to-noise ratio. Consequently, we must design filters that perform a derivation in one direction but also smooth the signal in all directions.

Smoothing is particularly effective in higher dimensional signals because smoothing in all directions perpendicular to the direction of the gradient does not blur the edge at all. Derivative filters that incorporate smoothing are also known as *regularized edge detectors* because they result in robust solutions for the ill-posed problem of estimating derivatives from discrete signals.

12.5.2 2×2 Cross-Smoothing Operator

The smallest cross-smoothing derivative operator has the following 2×2 masks

$$\boldsymbol{D}_x\boldsymbol{B}_y = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{D}_y\boldsymbol{B}_x = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \tag{12.51}$$

and the transfer functions

$$\begin{aligned} \hat{d}_x\hat{b}_y(\tilde{\boldsymbol{k}}) &= 2\mathrm{i}\sin(\pi\tilde{k}_x/2)\cos(\pi\tilde{k}_y/2) \\ \hat{d}_y\hat{b}_x(\tilde{\boldsymbol{k}}) &= 2\mathrm{i}\sin(\pi\tilde{k}_y/2)\cos(\pi\tilde{k}_x/2). \end{aligned} \tag{12.52}$$

There is nothing that can be optimized with this small filter mask. The filters $\boldsymbol{D}_x = [1\ -1]$ and $\boldsymbol{D}_y = [1\ -1]^T$ are not suitable to form a gradient operator, because $\boldsymbol{D}_x$ and $\boldsymbol{D}_y$ shift the convolution result by half a grid constant in the x and y directions, respectively.

The errors in the magnitude and direction of the gradient for small wave numbers are

$$e_m(\tilde{k},\phi) \approx -\frac{(\pi\tilde{k})^3}{24}\sin^2 2\phi + O(\tilde{k}^5). \tag{12.53}$$

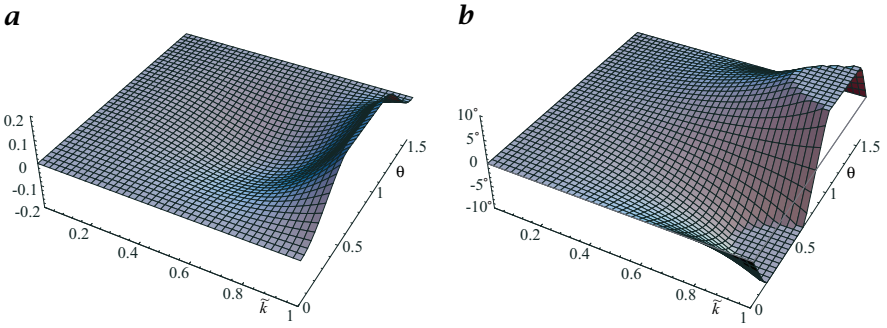

Figure 12.11: ***a*** *Anisotropy of the magnitude and* ***b*** *error in the direction of the gradient based on the Sobel edge detector Eq. (12.55). Parameters are the magnitude of the wave number (0 to 1) and the angle to the x axis (0 to π/2).*

$$e_\phi(\tilde{k},\phi) \approx -\frac{(\pi\tilde{k})^2}{48}\sin 4\phi + O(\tilde{k}^4). \tag{12.54}$$

The errors are significantly lower (a factor two for small wave numbers) as compared to the gradient computation based on the simple difference operator $\boldsymbol{D}_2 = 1/2\,[1\ 0\ -1]$ (Figs. 12.5 and 12.10), although the anisotropic terms occur in terms of the same order in Eqs. (12.28) and (12.29).

12.5.3 Sobel Edge Detector

The Sobel operator is the smallest difference filter with odd number of coefficients that averages the image in the direction perpendicular to the differentiation:

$$\boldsymbol{D}_{2x}\boldsymbol{B}_y^2 = \frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \quad \boldsymbol{D}_{2y}\boldsymbol{B}_x^2 = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}. \tag{12.55}$$

The errors in the magnitude and direction of the gradient based on Eq. (12.55) are shown in Fig. 12.11. The improvement over the simple symmetric derivative operator (Fig. 12.5) is similar to the 2×2 cross-smoothing difference operator (Fig. 12.10). A Taylor expansion in the wave number yields the same approximations (compare Eqs. (12.53) and (12.54)):

$$e_m(\tilde{k},\phi) \approx -\frac{(\pi\tilde{k})^3}{24}\sin^2 2\phi + O(\tilde{k}^5) \tag{12.56}$$

for the error of the magnitude and

$$e_\phi(\tilde{k},\phi) \approx -\frac{(\pi\tilde{k})^2}{48}\sin 4\phi + O(\tilde{k}^4) \tag{12.57}$$

for the direction of the gradient. A comparison with the corresponding equations for the simple difference filter Eqs. (12.28) and (12.29) shows that both the anisotropy and the angle error of the Sobel operator are a factor of two smaller. However, the error still increases with the square of the wave number. The error in the direction of the Sobel gradient is still up to 5° at a wave number of 0.5. For many applications, such a large error cannot be tolerated.

12.5.4 Derivatives of Gaussian

A well-known general class of regularized derivative filters is the class of derivates of a Gaussian smoothing filter. Such a filter was, e. g., used by Canny [18] for optimal edge detection and is also known as the *Canny edge detector*. On a discrete lattice this class of operators is best approximated by a derivative of a binomial operator (Section 11.4) as

$$^{(B,R)}\mathcal{D}_w = \mathcal{D}_{2w}\mathcal{B}^R \tag{12.58}$$

with nonsquare $(2R+3)\times(2R+1)^{W-1}$ W-dimensional masks and the transfer function

$$^{(B,R)}\hat{d}_w(\tilde{\boldsymbol{k}}) = \mathrm{i}\sin(\pi\tilde{k}_w)\prod_{w=1}^{W}\cos^{2R}(\pi\tilde{k}_w/2). \tag{12.59}$$

Surprisingly, this filter turns out to be a bad choice, because its anisotropy is the same as for the simple symmetric difference filter. This can be seen immediately for the direction of the gradient. The smoothing term is the same for both directions and thus cancels out in Eq. (12.19). The remaining terms are the same as for the symmetric difference filter.

In the same way, Sobel-type R^W-sized difference operators

$$^{R}S_w = \mathcal{D}_w\mathcal{B}_w^{R-1}\prod_{w'\neq w}\mathcal{B}_{w'}^{R} \tag{12.60}$$

with a $(2R+1)^W$ W-dimensional mask and the transfer function

$$^{R}\hat{S}_d(\tilde{k}) = \mathrm{i}\tan(\pi\tilde{k}_d/2)\prod_{w=1}^{W}\cos^{2R}(\pi\tilde{k}_d/2) \tag{12.61}$$

show the same anisotropy at the same wave number as the 3×3 Sobel operator.

12.5.5 Optimized Regularized Edge Detectors

It is easy to derive an optimized regularized derivative operator with a significantly lower error in the estimate of edges. A comparison of Eqs. (12.27) and (12.57) shows that the two filters have angle errors in opposite directions. Thus it appears that the Sobel operator performs too

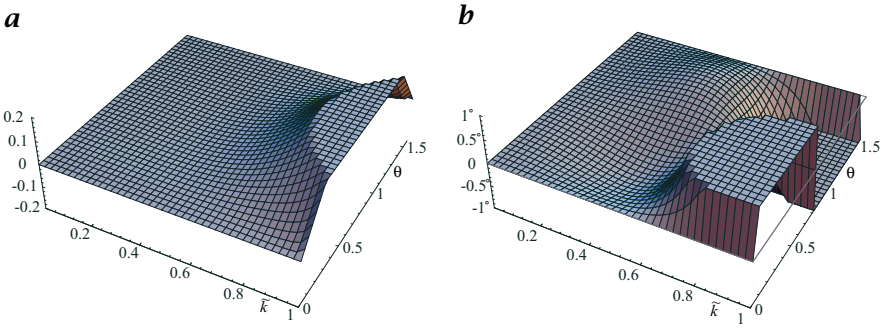

Figure 12.12: ***a*** *Anisotropy of the magnitude and* ***b*** *error in the direction of the gradient based on the optimized Sobel edge detector Eq. (12.62). Parameters are the magnitude of the wave number (0 to 1) and the angle to the x axis (0 to π/2).*

many cross-smoothings, while the symmetric difference operator performs too few. Consequently, we may suspect that a combination of both operators may result in a much lower error. Indeed, it is easy to reduce the cross-smoothing by increasing the central coefficient. Jähne et al. [85] show using a nonlinear optimization techniqe that the operators

$$
\begin{aligned}
1/4\boldsymbol{D}_{2x}(3\boldsymbol{B}_y^2 + \boldsymbol{I}) &= \frac{1}{32}\begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix}, \\
1/4\boldsymbol{D}_{2y}(3\boldsymbol{B}_x^2 + \boldsymbol{I}) &= \frac{1}{32}\begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix}
\end{aligned}
\tag{12.62}
$$

have a minimum angle error (Fig. 12.12). Similar optimizations are possible for larger-sized regularized derivative filters.

12.5.6 LoG and DoG Filter

Laplace filters tend to enhance the noise level in images considerably, because the transfer function is proportional to the wave number squared. Thus, a better edge detector may be found by first smoothing the image and then applying the Laplacian filter. This leads to a kind of regularized edge detection and to a class of filters called *Laplace of Gaussian* filters (*LoG* for short) or *Marr-Hildreth operator* [121].

In the discrete case, a LoG filter is approximated by first smoothing the image with a binomial mask and then applying the discrete Laplace filter. Thus we have the operator $\mathcal{LB}^p$ with the transfer function

$$
\hat{L}\hat{B}^p(\tilde{\boldsymbol{k}}) = -4\left[\sin^2(\pi\tilde{k}_x/2) + \sin^2(\pi\tilde{k}_y/2)\right]\cos^p(\pi\tilde{k}_x/2)\cos^p(\pi\tilde{k}_y/2). \tag{12.63}
$$

For small wave numbers, this transfer function can be approximated by

$$\hat{L}\hat{B}^p(\tilde{k},\phi) \approx -(\pi\tilde{k})^2 + \left[\frac{1}{16} + \frac{1}{8}p + \frac{1}{48}\cos(4\phi)\right](\pi\tilde{k})^4. \tag{12.64}$$

In Section 12.4.2 we saw that a Laplace filter can be even better approximated by operators of the type $\mathcal{B}^p - \mathcal{I}$. If additional smoothing is applied, this approximation for the Laplacian filter leads to the *difference of Gaussian* type of Laplace filter, or *DoG* filters:

$$4(\mathcal{B}^q - \mathcal{I})\mathcal{B}^p = 4(\mathcal{B}^{p+q} - \mathcal{B}^p). \tag{12.65}$$

The DoG filter $4(\mathcal{B}^{p+2} - \mathcal{B}^p)$ has the transfer function

$$\begin{aligned} 4(\hat{B}^{p+2} - \hat{B}^p)(\boldsymbol{k}) &= 4\cos^{p+2}(\pi\tilde{k}_x/2)\cos^{p+2}(\pi\tilde{k}_y/2) \\ &- 4\cos^{p}(\pi\tilde{k}_x/2)\cos^{p}(\pi\tilde{k}_y/2), \end{aligned} \tag{12.66}$$

which can be approximated for small wave numbers by

$$4(\hat{B}^{p+2} - \hat{B}^p)(\tilde{k},\phi) \approx -(\pi\tilde{k})^2 + \left[\frac{3}{32} + \frac{1}{8}p - \frac{1}{96}\cos(4\phi)\right](\pi\tilde{k})^4. \tag{12.67}$$

The transfer function of the LoG and DoG filters are compared in Fig. 12.13. It is obvious that the DoG filter is significantly more isotropic. A filter with even less deviation in the isotropy can be obtained by comparing Eqs. (12.64) and (12.67). The anisotropic $\cos 4\phi$ terms have different signs. Thus they can easily be compensated by a mix of LoG and DoG operators of the form 2/3DoG + 1/3LoG, which corresponds to the operator $(8/3\mathcal{B}^2 - 8/3\mathcal{I} - 1/3\mathcal{L})\mathcal{B}^p$.

DoG and LoG filter operators have some importance for the human visual system [120].

12.6 Edges in Multichannel Images‡

In multichannel images, it is significantly more difficult to analyze edges than to perform averaging, which was discussed in Section 11.8. The main difficulty is that the different channels may contain conflicting information about edges. In channel A, the gradient can point to a different direction than in channel B. The simple addition of the gradients in all channels

$$\sum_{p=1}^{P} \nabla g_p(\boldsymbol{x}) \tag{12.68}$$

is of no use here. It may happen that the gradients in two channels point in opposite directions and, thus, cancel each other. Then, the sum of the gradient over all channels would be zero, although the individual channels would

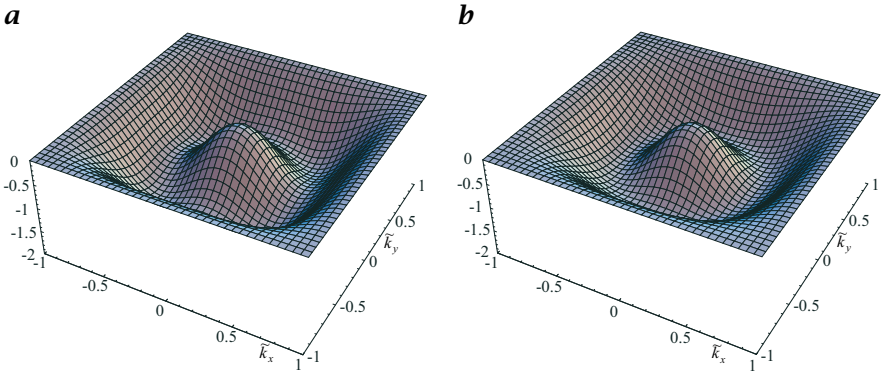

Figure 12.13: *Pseudo 3-D plot of the transfer function of* ***a*** *the LoG filter $\mathcal{LB}^2$ and* ***b*** *the DoG filter $4(\mathcal{B}^4 - \mathcal{B}^2)$.*

have non-zero gradients and we would be unable to distinguish this case from constant areas in both channels.

Thus, a more suitable measure of the total edge strength is the sum of the squared magnitudes of gradients in all channels:

$$\sum_{p=1}^{P} |\nabla g_p|^2 = \sum_{p=1}^{P} \sum_{w=1}^{W} \left(\frac{\partial g_p}{\partial x_w} \right)^2 . \tag{12.69}$$

While this expression gives a useful estimate of the overall edge strength, it still does not handle the problem of conflicting edge directions. An analysis of how edges are distributed in a W-dimensional multichannel image with P channels is possible with the following symmetric $W \times W$ matrix $\boldsymbol{S}$ (where W is the dimension of the image):

$$\boldsymbol{S} = \boldsymbol{J}^T \boldsymbol{J}, \tag{12.70}$$

where $\boldsymbol{J}$ is known as the *Jacobian matrix*. This $P \times W$ matrix is defined as

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_W} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_W} \\ \vdots & & \ddots & \vdots \\ \frac{\partial g_P}{\partial x_1} & \frac{\partial g_P}{\partial x_2} & \cdots & \frac{\partial g_P}{\partial x_W} \end{bmatrix} . \tag{12.71}$$

Thus the elements of the matrix $\boldsymbol{S}$ are

$$S_{kl} = \sum_{p=1}^{P} \frac{\partial g_p}{\partial x_k} \frac{\partial g_p}{\partial x_l} . \tag{12.72}$$

As $\boldsymbol{S}$ is a symmetric matrix, we can diagonalize it by a suitable coordinate transform. Then, we can write

$$\boldsymbol{S}' = \begin{bmatrix} \sum_p \left(\frac{\partial g_p}{\partial x'_1}\right)^2 & 0 & \cdots & 0 \\ 0 & \sum_p \left(\frac{\partial g_p}{\partial x'_2}\right)^2 & \ddots & 0 \\ 0 & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \sum_p \left(\frac{\partial g_p}{\partial x'_W}\right)^2 \end{bmatrix}. \tag{12.73}$$

In the case of an ideal edge, only one of the diagonal terms of the matrix will be non-zero. This is the direction perpendicular to the discontinuity. In all other directions it will be zero. Thus, $\boldsymbol{S}$ is a matrix of rank one in this case.

In contrast, if the edges in the different channels point randomly in all directions, all diagonal terms will be non-zero and equal. In this way, it is possible in principle to distinguish random changes by noise from coherent edges. The trace of the matrix $\boldsymbol{S}$

$$\text{trace}(\boldsymbol{S}) = \sum_{w=1}^{W} S_{ww} = \sum_{w=1}^{W} \sum_{p=1}^{P} \left(\frac{\partial g_p}{\partial x_w}\right)^2 \tag{12.74}$$

gives a measure of the edge strength which we have already defined in Eq. (12.69). It is independent of the orientation of the edge since the trace of a symmetric matrix is invariant to a rotation of the coordinate system.

12.7 Further Readings‡

A vast body of literature about edge detection is available. We will give here only a few selected references. The development of edge detection based on first-order difference filters can nicely be followed by a few key papers. Canny [18] formulated an optimal edge detector based on derivatives of the Gaussian, Deriche [30] introduced a fast recursive implementation of Canny's edge detector, Lanser and Eckstein [104] improved the isotropy of Deriche's recursive filter, and Jähne et al. [85] provide a nonlinear optimization strategy for edge detectors with optimal isotropy. Edge detection based on second-order difference (zero crossings) was strongly influenced by biological vision. The pioneering work is described by Marr and Hildreth [121] and Marr [120]. More recent work towards unified frameworks for neighborhood operators can be found in Koenderink and van Doorn [101] and Danielsson et al. [24].

13 Simple Neighborhoods

13.1 Introduction

In the last two chapters we became acquainted with neighborhood operations for performing averaging and detecting edges. In fact, we only studied the very simplest structures in a local neighborhood: constant areas and discontinuities. However, a local neighborhood could also contain patterns. In this chapter, we discuss the simplest class of such patterns, which we will call simple neighborhoods. As an introduction, we examine what types of simple patterns can be used to make an object distinguishable from a background for the human visual system.

Our visual system can easily recognize objects that do not differ from a background by their mean gray value but only by the orientation or scale of a pattern, as demonstrated in Fig. 13.1. To perform this recognition task with a digital image processing system, we need operators that determine the orientation and the scale of the pattern. After such an operation, a gray scale image is converted into a *feature image*. In the feature image, we can distinguish patterns that differ by orientation or scale in the same way we can distinguish gray values.

We denote local neighborhoods that can be described by an orientation as simple neighborhoods. The development of suitable operators for orientation and scale is an important and necessary requirement for analysis of more complex structures. It is interesting to observe that the meaning of one and the same local structure may be quite different, as illustrated in Fig. 13.2 for 2-D images:

- In the simplest case, the observed scene consists of objects and a background with uniform radiance (Fig. 13.2a). Then, a gray value change in a local neighborhood indicates that an edge of an object is encountered and the analysis of orientation yields the orientation of the edge.
- In Fig. 13.2b, the objects differ from the background by the orientation of the *texture*. Now, the local spatial structure does not indicate an edge but characterizes the texture of the objects. The analysis of texture will be discussed in Chapter 15.
- In image sequences, the local structure in the space-time domain is determined by motion, as illustrated by Fig. 13.2c for a 2-D space-

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2

Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

Figure 13.1: *An object can be distinguished from the background because it differs in* ***a*** *gray value,* ***b*** *the orientation of a pattern, or* ***c*** *the scale of a pattern.*

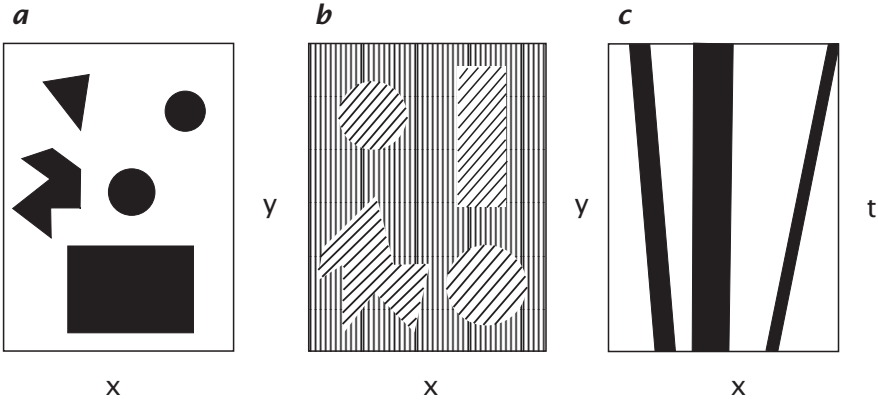

Figure 13.2: *Three different interpretations of local structures in 2-D images:* ***a*** *edge between uniform object and background;* ***b*** *orientation of pattern;* ***c*** *orientation in a 2-D space-time image indicating the velocity of 1-D objects.*

time image. Motion is an important feature, just like any other, for identifying objects and will be treated in detail in Chapter 14.

Although the three examples refer to entirely different image data, they have in common that the local structure is characterized by an orientation, i. e., the gray values change locally only in one direction. In this sense, the concept of orientation is an extension of the concept of edges.

13.2 Properties of Simple Neighborhoods

13.2.1 Representation in the Spatial Domain

The mathematical description of a local neighborhood is best done with continuous functions. This approach has two significant advantages. First, it is much easier to formulate the concepts and to study their properties analytically. As long as the corresponding discrete image satisfies the sampling theorem, all the results derived from continuous functions

remain valid, as the sampled image is an exact representation of the continuous gray value function. Second, we can now distinguish between errors inherent to the chosen approach and those that are only introduced by the discretization.

A local neighborhood with ideal local orientation is characterized by the fact that the gray value only changes in one direction. In all other directions it is constant. As the gray values are constant along lines, local orientation is also denoted as *linear symmetry* [8]. More recently, the term *simple neighborhood* has been coined by Granlund and Knutsson [57]. If we orient the coordinate system along the principal directions, the gray values become a 1-D function of only one coordinate. Generally, we will denote the direction of local orientation with a unit vector $\bar{\boldsymbol{n}}$ perpendicular to the lines of constant gray values. Then, a simple neighborhood is mathematically represented by

$$g(\boldsymbol{x}) = g(\boldsymbol{x}^T\bar{\boldsymbol{n}}), \tag{13.1}$$

where we denote the scalar product simply by $\boldsymbol{x}^T\bar{\boldsymbol{n}}$. We will use this simplified notation throughout this chapter. Equation (13.1) is also valid for image data with more than two dimensions. The projection of the vector $\boldsymbol{x}$ onto the unit vector $\bar{\boldsymbol{n}}$ makes the gray values depend only on a scalar quantity, the coordinate in the direction of $\bar{\boldsymbol{n}}$ (Fig. 13.3). It is easy to verify that this representation is correct by computing the gradient:

$$\nabla g(\boldsymbol{x}^T\bar{\boldsymbol{n}}) = \begin{bmatrix} \frac{\partial g(\boldsymbol{x}^T\bar{\boldsymbol{n}})}{\partial x_1} \\ \cdots \\ \frac{\partial g(\boldsymbol{x}^T\bar{\boldsymbol{n}})}{\partial x_W} \end{bmatrix} = \begin{bmatrix} \bar{n}_1 g'(\boldsymbol{x}^T\bar{\boldsymbol{n}}) \\ \cdots \\ \bar{n}_W g'(\boldsymbol{x}^T\bar{\boldsymbol{n}}) \end{bmatrix} = \bar{\boldsymbol{n}} g'(\boldsymbol{x}^T\bar{\boldsymbol{n}}). \tag{13.2}$$

With g' we denote the derivative of g with respect to the scalar variable $\boldsymbol{x}^T\bar{\boldsymbol{n}}$. In the hyperplane perpendicular to the gradient, the values remain locally constant. Equation Eq. (13.2) proves that the gradient lies in the direction of $\bar{\boldsymbol{n}}$.

13.2.2 Representation in the Fourier Domain

A simple neighborhood also has a special form in Fourier space. In order to derive it, we first assume that the whole image is described by Eq. (13.1), i. e., $\bar{\boldsymbol{n}}$ does not depend on the position. Then — from the very fact that a simple neighborhood is constant in all directions except $\bar{\boldsymbol{n}}$ — we infer that the Fourier transform must be confined to a line. The direction of the line is given by $\bar{\boldsymbol{n}}$:

$$g(\boldsymbol{x}^T\bar{\boldsymbol{n}}) \quad \circ\!\!-\!\!\bullet \quad \hat{g}(k)\delta(\boldsymbol{k} - \bar{\boldsymbol{n}}(\boldsymbol{k}^T\bar{\boldsymbol{n}})), \tag{13.3}$$

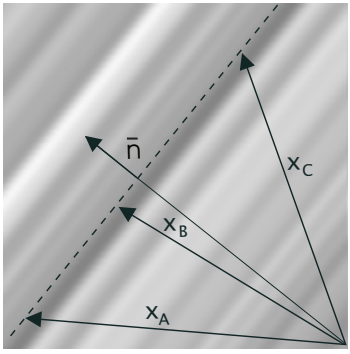

Figure 13.3: *Illustration of a linear symmetric or simple neighborhood. The gray values depend only on a coordinate given by a unit vector* $\bar{\boldsymbol{n}}$.

where k denotes the coordinate in the Fourier domain in the direction of $\bar{\boldsymbol{n}}$. The argument in the δ function is only zero when $\boldsymbol{k}$ is parallel to $\bar{\boldsymbol{n}}$. In a second step, we now restrict Eq. (13.3) to a local neighborhood by multiplying $g(\boldsymbol{x}^T\bar{\boldsymbol{n}})$ with a window function $w(\boldsymbol{x}-\boldsymbol{x}_0)$ in the spatial domain. Thus, we select a local neighborhood around $\boldsymbol{x}_0$. The size and shape of the neighborhood is determined by the window function. A window function that gradually decreases to zero diminishes the influence of pixels as a function of their distance from the outer pixel. Multiplication in the space domain corresponds to a convolution in the Fourier domain (Section 2.3). Thus,

$$w(\boldsymbol{x}-\boldsymbol{x}_0)\cdot g(\boldsymbol{x}^T\bar{\boldsymbol{n}}) \quad \circ\!\!-\!\!\bullet \quad \hat{w}(\boldsymbol{k}) * \hat{g}(k)\delta(\boldsymbol{k}-\bar{\boldsymbol{n}}(\boldsymbol{k}^T\bar{\boldsymbol{n}})), \quad (13.4)$$

where $\hat{w}(\boldsymbol{k})$ is the Fourier transform of the window function.

The limitation to a local neighborhood, thus, blurs the line in Fourier space to a "sausage-like" shape. Because of the reciprocity of scales between the two domains, its thickness is inversely proportional to the size of the window. From this elementary relation, we can already conclude qualitatively that the accuracy of the orientation estimate is directly related to the ratio of the window size to the wavelength of the smallest structures in the window.

13.2.3 Vector Representation of Local Neighborhoods

For an appropriate representation of simple neighborhoods, it is first important to distinguish *orientation* from *direction*. The direction is defined over the full angle range of 2π (360°). Two vectors that point in opposite directions, i. e., differ by 180°, are different. The gradient vector, for example, always points into the direction into which the gray values are increasing. With respect to a bright object on a dark background, this means that the gradient at the edge is pointing towards the

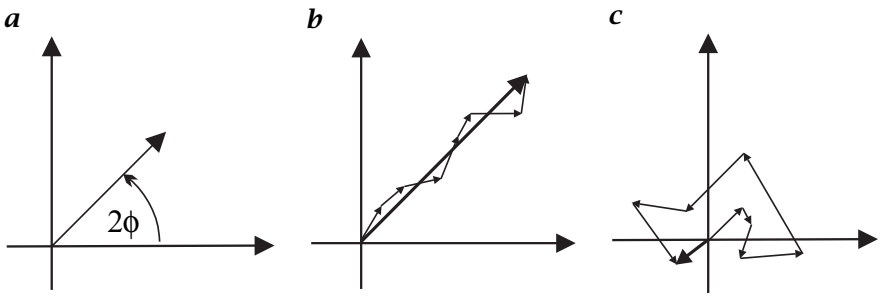

Figure 13.4: *Representation of local orientation as a vector:* ***a*** *the orientation vector;* ***b*** *averaging of orientation vectors from a region with homogeneous orientation;* ***c*** *same for a region with randomly distributed orientation.*

object. In contrast, to describe the direction of a local neighborhood, an angle range of 360° makes no sense. We cannot distinguish between patterns that are rotated by 180°. If a pattern is rotated by 180°, it still has the same direction. Thus, the direction of a simple neighborhood is different from the direction of a gradient. While for the edge of an object, gradients pointing in opposite directions are conflicting and inconsistent, for the direction of a simple neighborhood this is consistent information.

In order to distinguish the two types of "directions", we will speak of *orientation* in all cases where an angle range of only 180° is required. Orientation is still, of course, a *cyclic* quantity. Increasing the orientation beyond 180° flips it back to 0°. Therefore, an appropriate representation of orientation requires an angle doubling.

After this discussion of the principles of representing orientation, we are ready to think about an appropriate representation of simple neighborhoods. Obviously, a scalar quantity with just the doubled orientation angle is not appropriate. It seems to be useful to add a certainty measure that describes how well the neighborhood approximates a simple neighborhood. The scalar quantity and the certainty measure can be put together to form a vector. We set the magnitude of the vector to the certainty measure and the direction of the vector to the doubled orientation angle (Fig. 13.4a). This vector representation of orientation has two significant advantages.

First, it is more suitable for further processing than a separate representation of the orientation by two scalar quantities. Take, for example, averaging. Vectors are summed up by chaining them together, and the resulting sum vector is the vector from the starting point of the first vector to the end point of the last vector (Fig. 13.4b). The weight of an individual vector in the vector sum is given by its length. In this way, the certainty of the orientation measurement is adequately taken into

account. The vectorial representation of local orientation shows suitable averaging properties. In a region with homogeneous orientation the vectors line up to a large vector (Fig. 13.4b), i. e., a certain orientation estimate. In a region with randomly distributed orientation, however, the resulting vector remains small, indicating that no significant local orientation is present (Fig. 13.4c).

Second, it is difficult to display orientation as a gray scale image. While orientation is a cyclic quantity, the gray scale representation shows an unnatural jump between the smallest angle and the largest one. This jump dominates the appearance of the orientation images and, thus, does not give a good impression of the orientation distribution. The orientation vector can be well represented, however, as a color image. It appears natural to map the certainty measure onto the luminance and the orientation angle as the hue of the color. Our attention is then drawn to the bright parts in the images where we can distinguish the colors well. The darker a color is, the more difficult it gets to distinguish the different colors visually. In this way, our visual impression coincides with the orientation information in the image.

13.3 First-Order Tensor Representation†

13.3.1 The Structure Tensor

The vectorial representation discussed in Section 13.2.3 is incomplete. Although it is suitable for representing the orientation of simple neighborhoods, it cannot distinguish between neighborhoods with constant values and isotropic orientation distribution (e. g., uncorrelated noise). Both cases result in an orientation vector with zero magnitude.

Therefore, it is obvious that an adequate representation of gray value changes in a local neighborhood must be more complex. Such a representation should be able to determine a unique orientation (given by a unit vector $\bar{\boldsymbol{n}}$) and to distinguish constant neighborhoods from neighborhoods without local orientation.

A suitable representation can be introduced by the following optimization strategy to determine the orientation of a simple neighborhood. The optimum orientation is defined as the orientation that shows the least deviations from the directions of the gradient. A suitable measure for the deviation must treat gradients pointing in opposite directions equally. The squared scalar product between the gradient vector and the unit vector representing the local orientation $\bar{\boldsymbol{n}}$ meets this criterion:

$$(\nabla g^T \bar{\boldsymbol{n}})^2 = |\nabla g|^2 \cos^2 \left(\angle(\nabla g, \bar{\boldsymbol{n}})\right). \tag{13.5}$$

This quantity is proportional to the cosine squared of the angle between the gradient vector and the orientation vector and is thus maximal

when ∇g and $\bar{\boldsymbol{n}}$ are parallel or antiparallel, and zero if they are perpendicular to each other. Therefore, the following integral is maximized in a W-dimensional local neighborhood:

$$\int w(\boldsymbol{x}-\boldsymbol{x}')\left(\nabla g(\boldsymbol{x}')^T\bar{\boldsymbol{n}}\right)^2 \mathrm{d}^W x', \tag{13.6}$$

where the window function w determines the size and shape of the neighborhood around a point $\boldsymbol{x}$ in which the orientation is averaged. The maximization problem must be solved for each point $\boldsymbol{x}$. Equation Eq. (13.6) can be rewritten in the following way:

$$\bar{\boldsymbol{n}}^T\boldsymbol{J}\bar{\boldsymbol{n}} \rightarrow \text{maximum} \tag{13.7}$$

with

$$\boldsymbol{J} = \int w(\boldsymbol{x}-\boldsymbol{x}')\left(\nabla g(\boldsymbol{x}')\nabla g(\boldsymbol{x}')^T\right)\mathrm{d}^W x',$$

where $\nabla g \nabla g^T$ denotes an outer (Cartesian) product. The components of this symmetric $W \times W$ matrix are

$$J_{pq}(\boldsymbol{x}) = \int_{-\infty}^{\infty} w(\boldsymbol{x}-\boldsymbol{x}')\left(\frac{\partial g(\boldsymbol{x}')}{\partial x_p'}\frac{\partial g(\boldsymbol{x}')}{\partial x_q'}\right)\mathrm{d}^W x'. \tag{13.8}$$

These equations indicate that a tensor is an adequate first-order representation of a local neighborhood. The term first-order has a double meaning. First, only first-order derivatives are involved. Second, only simple neighborhoods can be described in the sense that we can analyze in which direction(s) the gray values change. More complex structures such as structures with multiple orientations cannot be distinguished.

The complexity of Eqs. (13.7) and (13.8) somewhat obscures their simple meaning. The tensor is symmetric. By a rotation of the coordinate system, it can be brought into a diagonal form. Then, Eq. (13.7) reduces in the 2-D case to

$$J' = [\bar{n}_1', \bar{n}_2']\begin{bmatrix} J_{11}' & 0 \\ 0 & J_{22}' \end{bmatrix}\begin{bmatrix} \bar{n}_1' \\ \bar{n}_2' \end{bmatrix} \quad \rightarrow \quad \text{maximum}. \tag{13.9}$$

A unit vector $\bar{\boldsymbol{n}}' = [\cos\phi \;\; \sin\phi]$ in the direction ϕ gives the values

$$J' = J_{11}'\cos^2\phi + J_{22}'\sin^2\phi.$$

Without loss of generality, we assume that $J_{11}' \geq J_{22}'$. Then, it is obvious that the unit vector $\bar{\boldsymbol{n}}' = [1\;0]^T$ maximizes Eq. (13.9). The maximum value is J_{11}'. In conclusion, this approach not only yields a tensor representation for the local neighborhood but also shows the way to determine the orientation. Essentially, we have to solve what is known as

Table 13.1: *Eigenvalue classification of the structure tensor in 2-D images.*

Condition	rank($\boldsymbol{J}$)	Description
$\lambda_1 = \lambda_2 = 0$	0	Both eigenvalues are zero. The mean squared magnitude of the gradient $(\lambda_1 + \lambda_2)$ is zero. The local neighborhood has constant values.
$\lambda_1 > 0, \lambda_2 = 0$	1	One eigenvalue is zero. The values do not change in the direction of the corresponding eigenvector. The local neighborhood is a simple neighborhood with ideal orientation.
$\lambda_1 > 0, \lambda_2 > 0$	2	Both eigenvalues are unequal to zero. The gray values change in all directions. In the special case of $\lambda_1 = \lambda_2$, we speak of an isotropic gray value structure as it changes equally in all directions.

an *eigenvalue problem*. The eigenvalues λ_w and eigenvectors $\boldsymbol{e}_w$ of a $W \times W$ matrix are defined by:

$$\boldsymbol{J}\boldsymbol{e}_w = \lambda_w \boldsymbol{e}_w. \tag{13.10}$$

An eigenvector $\boldsymbol{e}_w$ of $\boldsymbol{J}$ is thus a vector that is not turned in direction by multiplication with the matrix $\boldsymbol{J}$ but is only multiplied by a scalar factor, the eigenvalue λ_w. This implies that the structure tensor becomes diagonal in a coordinate system that is spanned by the eigenvectors Eq. (13.9). For our further discussion it is important to keep the following basic facts about the eigenvalues of a symmetric matrix in mind:

1. The eigenvalues are all real and non-negative.
2. The eigenvectors form an orthogonal basis.

According to the *maximization problem* formulated here, the eigenvector to the maximum eigenvalue gives the orientation of the local neighborhood.

13.3.2 Classification of Eigenvalues

The power of the tensor representation becomes apparent if we classify the eigenvalues of the structure tensor. The classifying criterion is the number of eigenvalues that are zero. If an eigenvalue is zero, this means that the gray values in the direction of the corresponding eigenvector do not change. The number of zero eigenvalues is also closely related to the rank of a matrix. The *rank* of a matrix is defined as the dimension of the subspace for which $\boldsymbol{J}\boldsymbol{k} \neq \boldsymbol{0}$. The space for which is $\boldsymbol{J}\boldsymbol{k} = \boldsymbol{0}$ is denoted as the *null space*. The dimension of the null space is the dimension of the matrix minus the rank of the matrix and equal to the number of zero

Table 13.2: *Eigenvalue classification of the structure tensor in 3-D (volumetric) images.*

Condition	rank($\boldsymbol{J}$)	Description
$\lambda_1 = \lambda_2 = \lambda_3 = 0$	0	The gray values do not change in any direction; constant neighborhood.
$\lambda_1 > 0, \lambda_2 = \lambda_3 = 0$	1	The gray values change only in one direction. This direction is given by the eigenvector to the non-zero eigenvalue. The neighborhood includes a boundary between two objects or a layered texture. In a space-time image, this means a constant motion of a spatially oriented pattern ("planar wave").
$\lambda_1 > 0, \lambda_2 > 0, \lambda_3 = 0$	2	The gray values change in two directions and are constant in a third. The eigenvector to the zero eigenvalue gives the direction of the constant gray values.
$\lambda_1 > 0, \lambda_2 > 0, \lambda_3 > 0$	3	The gray values change in all three directions.

eigenvalues. We will perform an analysis of the eigenvalues for two and three dimensions. In two and three dimensions, we can distinguish the cases summarized in Tables 13.1 and 13.2, respectively.

In practice, it will not be checked whether the eigenvalues are zero but below a critical threshold that is determined by the noise level in the image.

13.3.3 Orientation Vector

With the simple convolution and point operations discussed in the previous section, we computed the components of the structure tensor. In this section, we solve the eigenvalue problem to determine the orientation vector. In two dimensions, we can readily solve the eigenvalue problem. The orientation angle can be determined by rotating the inertia tensor into the principal axes coordinate system:

$$\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} J_{11} & J_{12} \\ J_{12} & J_{22} \end{bmatrix} \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix}.$$

Using the trigonometric identities $\sin 2\phi = 2\sin\phi\cos\phi$ and $\cos 2\phi = \cos^2\phi - \sin^2\phi$, the matrix multiplications result in

$$\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} =$$

$$\begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} J_{11}\cos\phi - J_{12}\sin\phi & J_{11}\sin\phi + J_{12}\cos\phi \\ -J_{22}\sin\phi + J_{12}\cos\phi & J_{22}\cos\phi + J_{12}\sin\phi \end{bmatrix} =$$

$$\begin{bmatrix} J_{11}\cos^2\phi + J_{22}\sin^2\phi - J_{12}\sin 2\phi & 1/2(J_{11}-J_{22})\sin 2\phi + J_{12}\cos 2\phi \\ 1/2(J_{11}-J_{22})\sin 2\phi + J_{12}\cos 2\phi & J_{11}\sin^2\phi + J_{22}\cos^2\phi + J_{12}\sin 2\phi \end{bmatrix}$$

Now we can compare the matrix coefficients on the left and right side of the equation. Because the matrices are symmetric, we have three equations with three unknowns, ϕ, λ_1, and λ_2. Although the equation system is nonlinear, it can readily be solved for ϕ.

A comparison of the off-diagonal elements on both sides of the equation

$$1/2(J_{11} - J_{22})\sin 2\phi + J_{12}\cos 2\phi = 0 \tag{13.11}$$

yields the orientation angle as

$$\tan 2\phi = \frac{2J_{12}}{J_{22} - J_{11}}. \tag{13.12}$$

Without defining any prerequisites, we have obtained the anticipated angle doubling for orientation. Since $\tan 2\phi$ is gained from a quotient, we can regard the dividend as the y and the divisor as the x component of a vector and can form the *orientation vector* $\boldsymbol{o}$, as introduced by Granlund [56]:

$$\boldsymbol{o} = \begin{bmatrix} J_{22} - J_{11} \\ 2J_{12} \end{bmatrix}. \tag{13.13}$$

The argument of this vector gives the orientation angle and the magnitude a certainty measure for local orientation.

The result of Eq. (13.13) is remarkable in that the computation of the components of the orientation vector from the components of the orientation tensor requires just one subtraction and one multiplication by two. As these components of the orientation vector are all we need for further processing steps we do not need the orientation angle or the magnitude of the vector. Thus, the solution of the eigenvalue problem in two dimensions is trivial.

13.3.4 Coherency

The orientation vector reduces local structure to local orientation. From three independent components of the symmetric tensor still only two are

used. When we fail to observe an orientated structure in a neighborhood, we do not know whether no gray value variations or distributed orientations are encountered. This information is included in the not yet used component of the tensor, $J_{11} + J_{22}$, which gives the mean square magnitude of the gradient. Consequently, a well-equipped structure operator needs to include also the third component. A suitable linear combination is

$$\boldsymbol{s} = \begin{bmatrix} J_{11} + J_{22} \\ J_{22} - J_{11} \\ 2J_{12} \end{bmatrix}. \tag{13.14}$$

This structure operator contains the two components of the orientation vector and, as an additional component, the mean square magnitude of the gradient, which is a rotation-invariant parameter. Comparing the latter with the magnitude of the orientation vector, a constant gray value area and an isotropic gray value structure without preferred orientation can be distinguished. In the first case, both squared quantities are zero, in the second only the magnitude of the orientation vector. In the case of a perfectly oriented pattern, both quantities are equal. Thus their ratio seems to be a good *coherency measure* c_c for local orientation:

$$c_c = \frac{\sqrt{(J_{22} - J_{11})^2 + 4J_{12}^2}}{J_{11} + J_{22}} = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}. \tag{13.15}$$

The coherency ranges from 0 to 1. For ideal local orientation ($\lambda_2 = 0$, $\lambda_1 > 0$) it is one, for an isotropic gray value structure ($\lambda_1 = \lambda_2 > 0$) it is zero.

13.3.5 Color Coding of the 2-D Structure Tensor

In Section 13.2.3 we discussed a color representation of the orientation vector. The question is whether it is also possible to represent the structure tensor adequately as a color image. A symmetric 2-D tensor has three independent pieces of information Eq. (13.14), which fit well to the three degrees of freedom available to represent color, for example luminance, hue, and saturation.

A color represention of the structure tensor requires only two slight modifications as compared to the color representation for the orientation vector. First, instead of the length of the orientation vector, the squared magnitude of the gradient is mapped onto the intensity. Second, the coherency measure Eq. (13.15) is used as the saturation. In the color representation for the orientation vector, the saturation is always one. The angle of the orientation vector is still represented as the hue.

In practice, a slight modification of this color representation is useful. The squared magnitude of the gradient shows variations too large to be displayed in the narrow dynamic range of a display screen with only 256

luminance levels. Therefore, a suitable normalization is required. The basic idea of this normalization is to compare the squared magnitude of the gradient with the noise level. Once the gradient is well above the noise level it is regarded as a significant piece of information. This train of thoughts suggests the following normalization for the intensity I:

$$I = \frac{J_{11} + J_{22}}{(J_{11} + J_{22}) + \gamma\sigma_n^2}, \tag{13.16}$$

where σ_n is an estimate of the standard deviation of the noise level. This normalization provides a rapid transition of the luminance from one, when the magnitude of the gradient is larger than σ_n, to zero when the gradient is smaller than σ_n. The factor γ is used to optimize the display.

13.3.6 Implementation

The structure tensor (Section 13.3.1) or the inertia tensor (Section 13.3.7) can be computed straightforwardly as a combination of *linear convolution* and *nonlinear point operations*. The partial derivatives in Eqs. (13.8) and (13.25) are approximated by discrete derivative operators. The integration weighted with the window function is replaced by a convolution with a smoothing filter which has the shape of the window function. If we denote the discrete partial derivative operator with respect to the coordinate p by the operator $\mathcal{D}_p$ and the (isotropic) smoothing operator by $\mathcal{B}$, the local structure of a gray value image can be computed with the structure tensor operator

$$\mathcal{J}_{pq} = \mathcal{B}(\mathcal{D}_p \cdot \mathcal{D}_q). \tag{13.17}$$

The equation is written in an operator notation. Pixelwise multiplication is denoted by $\cdot$ to distinguish it from successive application of convolution operators. Equation Eq. (13.17) says, in words, that the $\mathcal{J}_{pq}$ component of the tensor is computed by convolving the image independently with $\mathcal{D}_p$ and $\mathcal{D}_q$, multiplying the two images pixelwise, and smoothing the resulting image with $\mathcal{B}$.

These operators are valid in images of any dimension $W \geq 2$. In a W-dimensional image, the structure tensor has $W(W+1)/2$ independent components, hence 3 in 2-D, 6 in 3-D, and 10 in 4-D images. These components are best stored in a multichannel image with $W(W+1)/2$ components.

The smoothing operations consume the largest number of operations. Therefore, a fast implementation must, in the first place, apply a fast smoothing algorithm. A fast algorithm can be established based on the general observation that higher-order features always show a lower resolution than the features they are computed from. This means that

the structure tensor can be stored on a coarser grid and thus in a smaller image. A convenient and appropriate subsampling rate is to reduce the scale by a factor of two by storing only every second pixel in every second row.

These procedures lead us in a natural way to multigrid data structures which are discussed in detail in Chapter 5. Multistep averaging is discussed in detail in Section 11.6.1.

Storing higher-order features on coarser scales has another significant advantage. Any subsequent processing is sped up simply by the fact that many fewer pixels have to be processed. A linear scale reduction by a factor of two results in a reduction in the number of pixels and the number of computations by a factor of 4 in two and 8 in three dimensions.

Figure 13.5 illustrates all steps to compute the structure tensor and derived quantities using the ring test pattern. This test pattern is particularly suitable for orientation analysis since it contains all kinds of orientations and wave numbers in one image.

The accuracy of the orientation angle strongly depends on the implementation of the derivative filters. The straightforward implementation of the algorithm using the standard derivative filter mask $1/2\,[1\ 0\ -1]$ (Section 12.3.3) or the *Sobel operator* (Section 12.5.3) results in surprisingly high errors (Fig. 13.6b), with a maximum error in the orientation angle of more than 7° at a wave number of $\tilde{k} = 0.7$. The error depends on both the wave number and the orientation of the local structure. For orientation angles in the direction of axes and diagonals, the error vanishes. The high error and the structure of the error map result from the transfer function of the derivative filter. The transfer function shows significant deviation from the transfer function for an ideal derivative filter for high wave numbers (Section 12.2). According to Eq. (13.12), the orientation angle depends on the ratio of derivatives. Along the axes, one of the derivatives is zero and, thus, no error occurs. Along the diagonals, the derivatives in the x and y directions are the same. Consequently, the error in both cancels in the ratio of the derivatives as well.

The error in the orientation angle can be significantly suppressed if better derivative filters are used. Figure 13.6 shows the error in the orientation estimate using two examples of the optimized Sobel operator (Section 12.5.5) and the least-squares optimized operator (Section 12.3.5).

The little extra effort in optimizing the derivative filters thus pays off in an accurate orientation estimate. A residual angle error of less than 0.5° is sufficient for almost all applications. The various derivative filters discussed in Sections 12.3 and 12.5 give the freedom to balance computational effort with accuracy.

An important property of any image processing algorithm is its *robustness*. This term denotes the sensitivity of an algorithm against noise. Two questions are important. First, how large is the error of the esti-

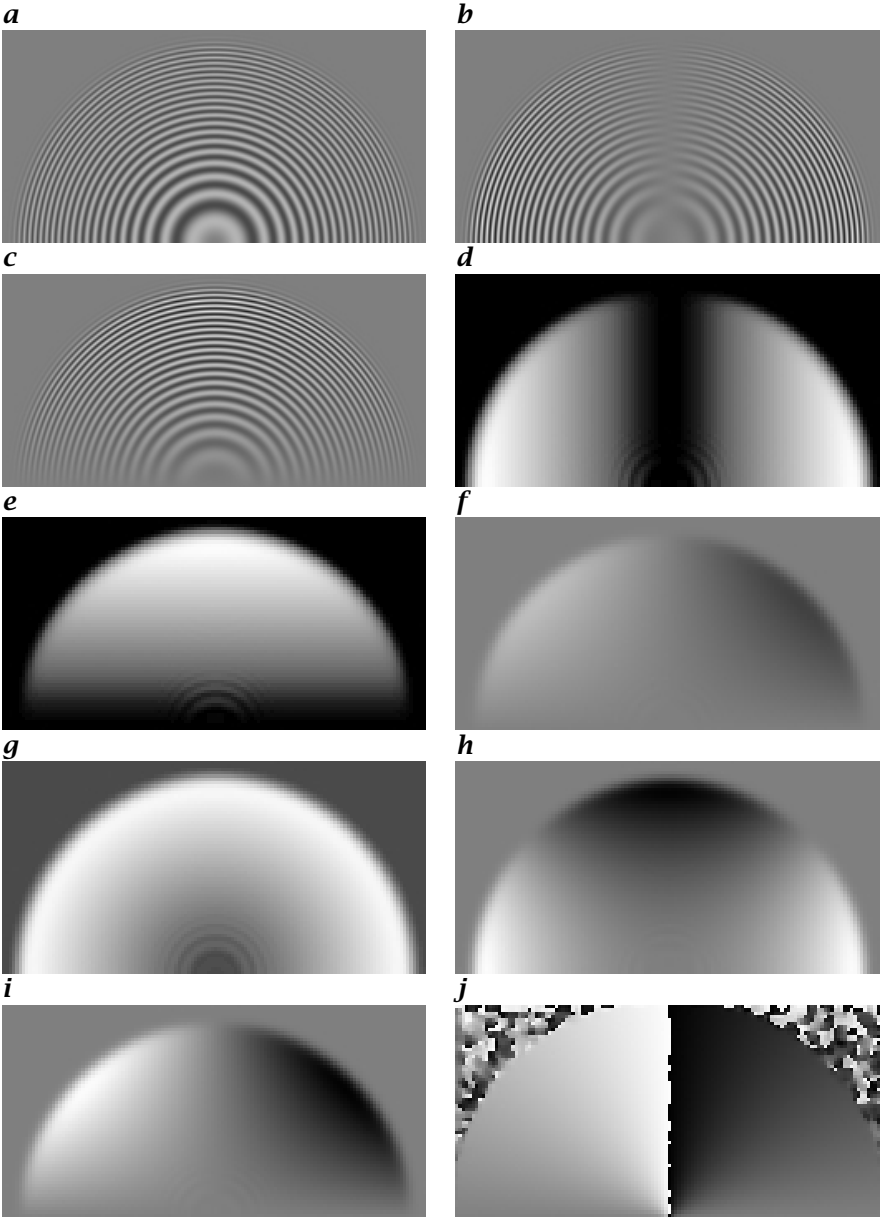

Figure 13.5: *Steps to compute the structure tensor:* **a** *original ring test pattern;* **b** *horizontal derivation* $\mathcal{D}_x$*;* **c** *vertical derivation* $\mathcal{D}_y$*;* **d–f** *averaged components for the structure tensor* $J_{11} = \mathcal{B}(\mathcal{D}_x \cdot \mathcal{D}_x), J_{22} = \mathcal{B}(\mathcal{D}_y \cdot \mathcal{D}_y), J_{12} = \mathcal{B}(\mathcal{D}_x \cdot \mathcal{D}_y)$*;* **g** *squared magnitude of gradient* $J_{11} + J_{22}$*;* **h** x *component of orientation vector* $J_{11} - J_{22}$*;* **i** y *component of orientation vector* $2J_{12}$*;* **j** *orientation angle from* $[-\pi/2, \pi/2]$ *mapped to a gray scale interval from [0, 255].*

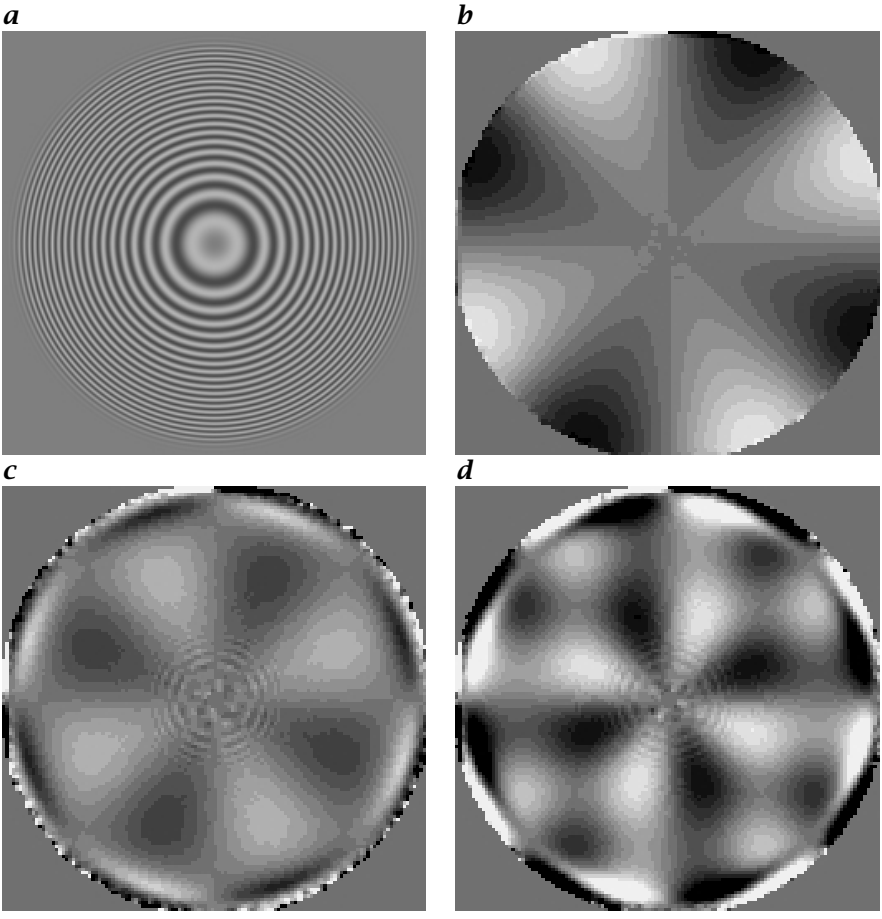

Figure 13.6: *Systematic errors for the orientation angle estimate using different derivative operators:* ***a*** *original ring test pattern with a maximum normalized wave number $\tilde{k} = 0.7$; error maps for* ***b*** *the Sobel operator (angle range $\pm 7°$ in 16 discrete steps),* ***c*** *the optimized Sobel operator, and* ***d*** *the least squares optimized operator (angle range $\pm 0.7°$ in 16 discrete steps) with $r = 3$.*

mated features in noisy images? To answer this question, the laws of statistics are used to study error propagation. In this context, noise makes the estimates only uncertain but not erroneous. The mean — if we make a sufficient number of estimates — is still correct. However, a second question arises. In noisy images an operator can also give results that are biased, i. e., the mean can show a significant deviation from the correct value. In the worst case, an algorithm can even become unstable and deliver meaningless results.

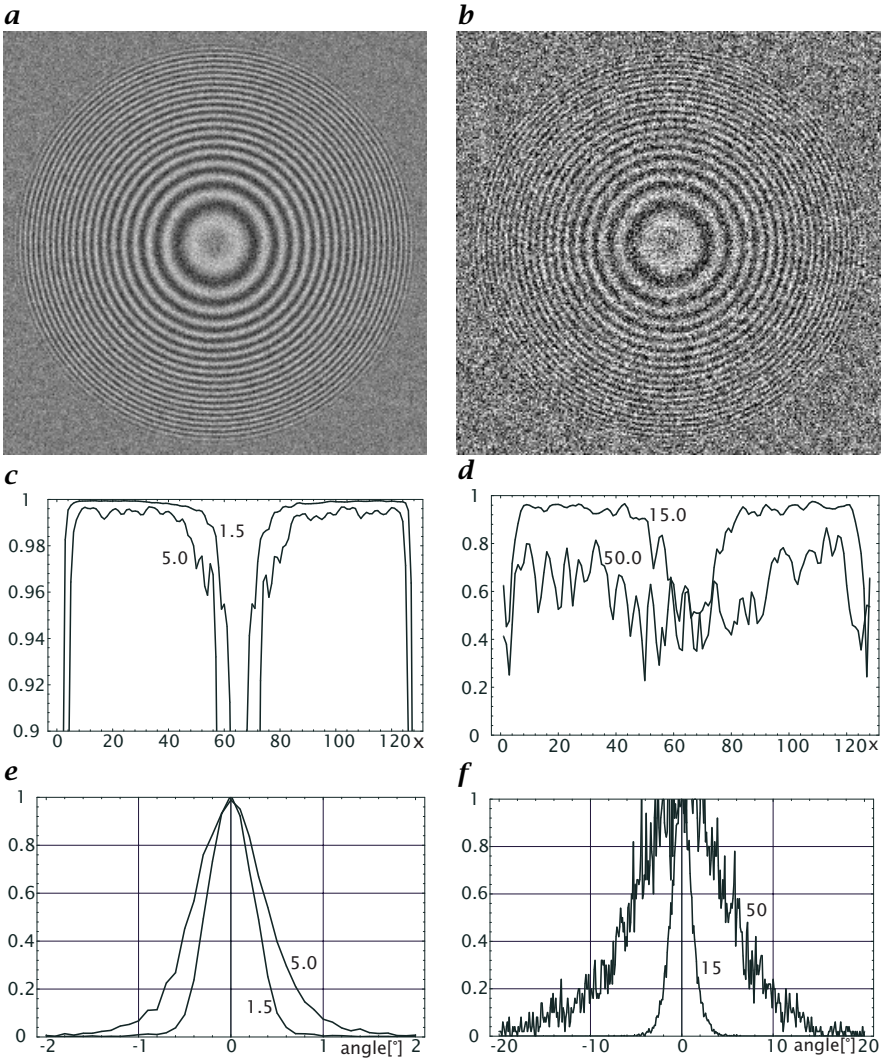

Figure 13.7: *Orientation analysis with a noisy ring test pattern using the optimized Sobel operator: ring pattern with amplitude 50, standard deviation of normal distributed noise* ***a*** *15, and* ***b*** *50;* ***c*** *and* ***d*** *radial cross section of the coherency measure for standard deviations of the noise level of 1.5 and 5, 15 and 50, respectively;* ***e*** *and* ***f*** *histograms of angle error for the same conditions.*

Figure 13.7 demonstrates that the estimate of orientation is also a remarkably robust algorithm. Even with a low signal-to-noise ratio, the orientation estimate is still correct if a suitable derivative operator is used. With increasing noise level, the coherency (Section 13.3.4) decreases and the statistical error of the orientation angle estimate increases (Fig. 13.7).

13.3.7 The Inertia Tensor‡

In this section, we discuss an alternative approach to describe the local structure in images. As a starting point, we consider what an ideally oriented gray value structure (Eq. (13.1)) looks like in the wave number domain. We can compute the Fourier transform of Eq. (13.1) more readily if we rotate the x_1 axis in the direction of $\bar{\boldsymbol{n}}$. Then the gray value function is constant in the x_2 direction. Consequently, the Fourier transform reduces to a δ line in the direction of $\bar{\boldsymbol{n}}$ (≻ R5).

It seems promising to determine local orientation in the Fourier domain, as all we have to compute is the orientation of the line on which the spectral densities are non-zero. Bigün and Granlund [8] devised the following procedure:

- Use a window function to select a small local neighborhood from an image.
- Fourier transform the windowed image. The smaller the selected window, the more blurred the spectrum will be (*uncertainty relation*, Theorem 7, p. 55). This means that even with an ideal local orientation we will obtain a rather band-shaped distribution of the spectral energy.
- Determine local orientation by fitting a straight line to the spectral density distribution. This yields the angle of the local orientation from the slope of the line.

The critical step of this procedure is fitting a straight line to the spectral densities in the Fourier domain. We cannot solve this problem exactly as it is generally *overdetermined*, but only minimize the measure of error. A standard error measure is the square of the magnitude of the vector (L_2 norm; see Eq. (2.74) in Section 2.4.1). When fitting a straight line, we minimize the sum of the squares of the distances of the data points to the line:

$$\int_{-\infty}^{\infty} d^2(\boldsymbol{k}, \bar{\boldsymbol{n}}) |\hat{g}(\boldsymbol{k})|^2 \mathrm{d}^W k \rightarrow \text{minimum}. \tag{13.18}$$

The distance function is abbreviated using $d(\boldsymbol{k}, \bar{\boldsymbol{n}})$. The integral runs over the whole wave number space; the wave numbers are weighted with the spectral density $|\hat{g}(\boldsymbol{k})|^2$. Equation (13.18) is not restricted to two dimensions, but is generally valid for local orientation or linear symmetry in a W-dimensional space. The distance vector $\boldsymbol{d}$ can be inferred from Fig. 13.8 to be

$$\boldsymbol{d} = \boldsymbol{k} - (\boldsymbol{k}^T \bar{\boldsymbol{n}}) \bar{\boldsymbol{n}}. \tag{13.19}$$

The square of the distance is then given by

$$|\boldsymbol{d}|^2 = |\boldsymbol{k} - (\boldsymbol{k}^T \bar{\boldsymbol{n}}) \bar{\boldsymbol{n}}|^2 = |\boldsymbol{k}|^2 - (\boldsymbol{k}^T \bar{\boldsymbol{n}})^2. \tag{13.20}$$

In order to express the distance more clearly as a function of the vector $\bar{\boldsymbol{n}}$, we rewrite it in the following manner:

$$|\boldsymbol{d}|^2 = \bar{\boldsymbol{n}}^T (\boldsymbol{I}(\boldsymbol{k}^T \boldsymbol{k}) - (\boldsymbol{k}\boldsymbol{k}^T)) \bar{\boldsymbol{n}}, \tag{13.21}$$

where $\boldsymbol{I}$ is the unit diagonal matrix. Substituting this expression into Eq. (13.18) we obtain

$$\bar{\boldsymbol{n}}^T \boldsymbol{J}' \bar{\boldsymbol{n}} \rightarrow \text{minimum}, \tag{13.22}$$

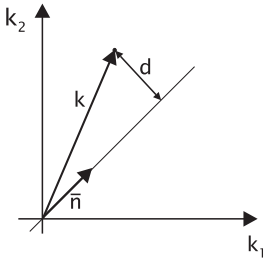

Figure 13.8: *Distance of a point in the wave number space from the line in the direction of the unit vector $\bar{\boldsymbol{n}}$.*

where $\boldsymbol{J}'$ is a symmetric tensor with the diagonal elements

$$J'_{pp} = \sum_{q \neq p} \int_{-\infty}^{\infty} k_q^2 |\hat{g}(\boldsymbol{k})|^2 \mathrm{d}^W k \tag{13.23}$$

and the off-diagonal elements

$$J'_{pq} = -\int_{-\infty}^{\infty} k_p k_q |\hat{g}(\boldsymbol{k})|^2 \mathrm{d}^W k, \quad p \neq q. \tag{13.24}$$

The tensor $\boldsymbol{J}'$ is analogous to a well-known physical quantity, the *inertia tensor*. If we replace the wave number coordinates by space coordinates and the spectral density $|\hat{g}(\boldsymbol{k})|^2$ by the specific density ρ, Eqs. (13.18) and (13.22) constitute the equation to compute the inertia of a rotary body rotating around the $\bar{\boldsymbol{n}}$ axis.

With this analogy, we can reformulate the problem of determining local orientation. We must find the axis about which the rotary body, formed from the spectral density in Fourier space, rotates with minimum inertia. This body might have different shapes. We can relate its shape to the different solutions we get for the eigenvalues of the inertia tensor and thus for the solution of the local orientation problem (Table 13.3).

We derived the inertia tensor approach in the Fourier domain. Now we will show how to compute the coefficients of the inertia tensor in the space domain.

The integrals in Eqs. (13.23) and (13.24) contain terms of the form

$$k_q^2 |\hat{g}(\boldsymbol{k})|^2 = |\mathrm{i} k_q \hat{g}(\boldsymbol{k})|^2$$

and

$$k_p k_q |\hat{g}(\boldsymbol{k})|^2 = \mathrm{i} k_p \hat{g}(\boldsymbol{k}) [\mathrm{i} k_q \hat{g}(\boldsymbol{k})]^*.$$

Integrals over these terms are *inner* or *scalar products* of the functions $\mathrm{i} k_p \hat{g}(\boldsymbol{k})$. Because the inner product is preserved under the Fourier transform (≻ R4), we can compute the corresponding integrals in the spatial domain as well. Multiplication of $\hat{g}(\boldsymbol{k})$ with $\mathrm{i} k_p$ in the wave number domain corresponds to performing

Table 13.3: *Eigenvalue classification of the structure tensor in 3-D (volumetric) images.*

Condition	Explanation
Ideal local orientation	The rotary body is a line. For a rotation around this line, the inertia vanishes. Consequently, the eigenvector to the eigenvalue zero coincides with the direction of the line. The other eigenvector is orthogonal to the line, and the corresponding eigenvalue is unequal to zero and gives the rotation axis for maximum inertia.
Isotropic gray value structure	In this case, the rotary body is a kind of flat isotropic disk. A preferred direction does not exist. Both eigenvalues are equal and the inertia is the same for rotations around all axes. We cannot find a minimum.
Constant gray values	The rotary body degenerates to a point at the origin of the wave number space. The inertia is zero for rotation around any axis. Therefore both eigenvalues vanish.

the first spatial derivative in the direction of x_p in the space domain:

$$\begin{aligned} J'_{pp}(\boldsymbol{x}) &= \sum_{q\neq p} \int_{-\infty}^{\infty} w(\boldsymbol{x}-\boldsymbol{x}') \left(\frac{\partial g}{\partial x_q}\right)^2 \mathrm{d}^W x' \\ J'_{pq}(\boldsymbol{x}) &= -\int_{-\infty}^{\infty} w(\boldsymbol{x}-\boldsymbol{x}') \frac{\partial g}{\partial x_p}\frac{\partial g}{\partial x_q} \mathrm{d}^W x'. \end{aligned} \tag{13.25}$$

In Eq. (13.25), we already included the weighting with the window function w to select a local neighborhood.

The structure tensor discussed in Section 13.3.1 Eq. (13.8) and the inertia tensor are closely related:

$$\boldsymbol{J}' = \mathrm{trace}(\boldsymbol{J})\boldsymbol{I} - \boldsymbol{J}. \tag{13.26}$$

From this relationship it is evident that both matrices have the same set of eigenvectors. The eigenvalues λ_p are related by

$$\lambda_p = \sum_{q=1}^{n} \lambda_q - \lambda'_p, \quad \lambda'_p = \sum_{q=1}^{n} \lambda_q - \lambda_p. \tag{13.27}$$

Consequently, we can perform the eigenvalue analysis with any of the two matrices. For the inertia tensor, the direction of local orientation is given by the minimum eigenvalue, but for the structure tensor it is given by the maximum eigenvalue.

13.3.8 Further Equivalent Approaches‡

In their paper on analyzing oriented patterns, Kass and Witkin [90] chose — at first glance — a completely different method. Yet it turns out to be equivalent to the tensor method, as will be shown in the following. They started with the idea of using *directional derivative* filters by differentiating a *difference of Gaussian filter* (*DoG*, Section 12.5.6) (written in operator notation)

$$\mathcal{R}(\Theta) = [\cos\Theta \;\; \sin\Theta] \begin{bmatrix} \mathcal{D}_x(\mathcal{B}_1 - \mathcal{B}_2) \\ \mathcal{D}_y(\mathcal{B}_1 - \mathcal{B}_2) \end{bmatrix} = [\cos\Theta \;\; \sin\Theta] \begin{bmatrix} \mathcal{R}_x \\ \mathcal{R}_y \end{bmatrix},$$

where $\mathcal{B}_1$ and $\mathcal{B}_2$ denote two Gaussian smoothing masks with different variances. The direction in which this directional derivative is maximal in a mean square sense gives the orientation normal to lines of constant gray values. This approach results in the following expression for the variance of the directional derivative:

$$\mathcal{V}(\Theta) = \mathcal{B}(\mathcal{R}(\Theta) \cdot \mathcal{R}(\Theta)). \tag{13.28}$$

The directional derivative is squared and then smoothed by a binomial filter. This equation can also be interpreted as the inertia of an object as a function of the angle. The corresponding inertia tensor has the form

$$\begin{bmatrix} \mathcal{B}(\mathcal{R}_y \cdot \mathcal{R}_y) & -\mathcal{B}(\mathcal{R}_x \cdot \mathcal{R}_y) \\ -\mathcal{B}(\mathcal{R}_x \cdot \mathcal{R}_y) & \mathcal{B}(\mathcal{R}_x \cdot \mathcal{R}_x) \end{bmatrix}. \tag{13.29}$$

Thus Kass and Witkin's approach is identical to the general inertia tensor method discussed in Section 13.3.7. They just used a special type of derivative filter.

Without being aware of either Bigün and Granlund [8] earlier or Knutsson [99] contemporary work, Rao and Schunck [147] and Rao [146] proposed the same structure tensor (denoting it as the moment tensor) as that we discussed in Section 13.3.1.

13.4 Local Wave Number and Phase‡

13.4.1 Phase‡

So far in this chapter we have discussed in detail the analysis of simple neighborhoods with respect to their orientation. In this section we proceed with another elementary property of simple neighborhoods. In Chapter 5 we stressed the importance of the scale for image processing. Thus we must not only ask in which directions the gray values change. We must also ask how fast the gray values change. This question leads us to the concept of the *local wave number*.

The key to determining the local wave number is the *phase* of the signal. As an introduction we discuss a simple example and consider the one-dimensional periodic signal

$$g(x) = g_0 \cos(kx). \tag{13.30}$$

The argument of the cosine function is known as the phase of the periodic signal:

$$\phi(x) = kx. \tag{13.31}$$

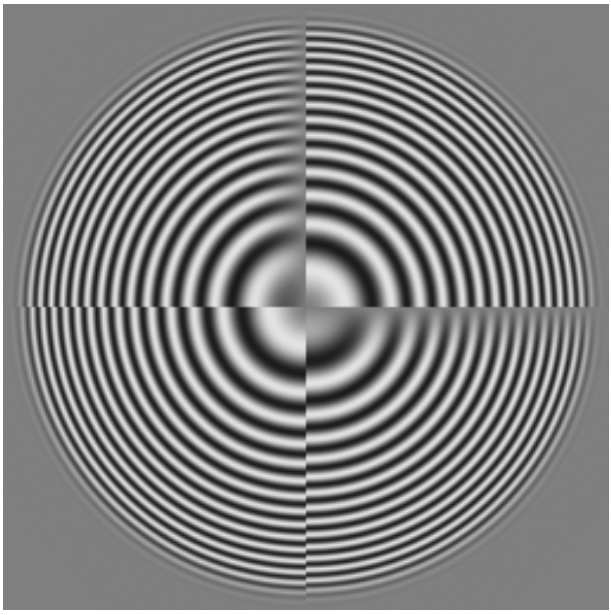

Figure 13.9: *Application of the Hilbert filter to the ring test pattern: upper left quadrant: in the horizontal direction; lower right quadrant: in the vertical direction.*

The equation shows that the phase is a linear function of the position and the wave number. Thus we obtain the wave number of the periodic signal by computing the first-order spatial derivative of the phase signal

$$\frac{\partial \phi(x)}{\partial x} = k. \tag{13.32}$$

These simple considerations re-emphasize the significant role of the phase in image processing that we discussed already in Section 2.3.6. We will discuss two related approaches for determining the phase of a signal, the *Hilbert transform* (Section 13.4.2) and the *quadrature filter* (Section 13.4.5) before we introduce efficient techniques to compute the local wave number from phase gradients.

13.4.2 Hilbert Transform and Hilbert Filter‡

In order to explain the principle of computing the phase of a signal, we take again the example of the simple periodic signal from the previous section. We suppose that an operator is available to delay the signal by a phase of 90°. This operator would convert the $g(x) = g_0 \cos(kx)$ signal into a $g'(x) = -g_0 \sin(kx)$ signal as illustrated in Fig. 13.9. Using both signals, the phase of $g(x)$ can be computed by

$$\phi(g(x)) = \arctan\left(\frac{-g'(x)}{g(x)}\right). \tag{13.33}$$

As only the ratio of $g'(x)$ and $g(x)$ goes into Eq. (13.33), the phase is indeed independent of amplitude. If we take the signs of the two functions $g'(x)$ and $g(x)$ into account, the phase can be computed over the full range of 360°.

Thus all we need to determine the phase of a signal is a linear operator that shifts the phase of a signal by 90°. Such an operator is known as the *Hilbert filter* $\boldsymbol{H}$ or *Hilbert operator* $\mathcal{H}$ and has the transfer function

$$\hat{h}(k) = \begin{cases} \mathrm{i} & k > 0 \\ 0 & k = 0 \\ -\mathrm{i} & k < 0 \end{cases}. \tag{13.34}$$

The magnitude of the transfer function is one as the amplitude remains unchanged. As the Hilbert filter has a purely imaginary transfer function, it must be of odd symmetry to generate a real-valued signal. Therefore positive wave numbers are shifted by 90° ($\pi/2$) and negative wave numbers by −90° ($-\pi/2$). A special situation is given for the wave number zero where the transfer function is also zero. This exception can be illustrated as follows. A signal with wave number zero is a constant. It can be regarded as a cosine function with infinite wave number sampled at the phase zero. Consequently, the Hilbert filtered signal is the corresponding sine function at phase zero, that is, zero.

Because of the discontinuity of the transfer function of the Hilbert filter at the origin, its point spread function is of infinite extent

$$h(x) = -\frac{1}{\pi x}. \tag{13.35}$$

The convolution with Eq. (13.35) can be written as

$$g_h(x) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{g(x')}{x' - x} \mathrm{d}x'. \tag{13.36}$$

This integral transform is known as the *Hilbert transform* [116].

Because the convolution mask of the Hilbert filter is infinite, it is impossible to design an exact discrete Hilbert filter for arbitrary signals. This is only possible if we restrict the class of signals to which it is applied. Thus the following approach is taken to design an effective implementation of a Hilbert filter.

First, the filter should precisely shift the phase by $\pi/2$. This requirement comes from the fact that we cannot afford an error in the phase because it includes the position information. A wave-number dependent phase shift would cause wave-number dependent errors. This requirement is met by any convolution kernel of odd symmetry.

Second, requirements for a magnitude of one can be relaxed if the Hilbert filter is applied to a bandpassed signal, e. g., the Laplace pyramid. Then, the Hilbert filter must only show a magnitude of one in the passband range of the bandpass filter used. This approach avoids the discontinuities in the transfer function at the wave number zero and thus results in finite-sized convolution kernels.

Optimized Hilbert filters are generated with the same least-squares techniques used above for interpolation filters (Section 10.6.6) and first-order derivative filters (Section 12.3.5).

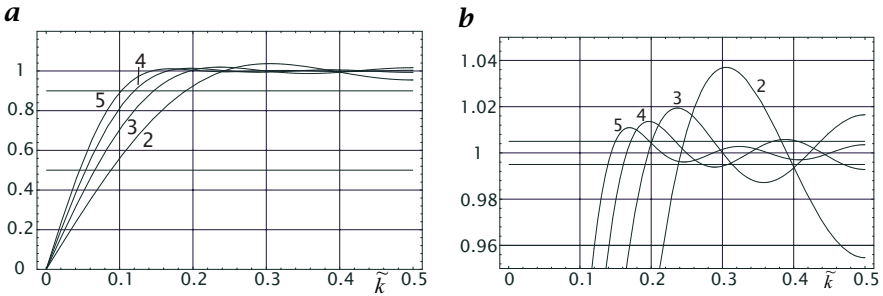

Figure 13.10: ***a*** *Transfer functions of a family of least-squares optimized Hilbert operators according to Eq. (13.37) for the four filter coefficients $R = 2, 3, 4, 5$.* ***b*** *sector of* ***a*** *to better show the deviations from an ideal Hilbert filter. As the filters are symmetric around $\tilde{k} = 0.5$ only a wave number range from 0–0.5 is shown.*

Because of the odd symmetry of the Hilbert filter, the following formulation is used

$$\hat{h}(\tilde{k}) = 2\mathrm{i}\sum_{v=1}^{R} h_v \sin\left((2v-1)\pi\tilde{k}\right). \tag{13.37}$$

Note that we have only used sine functions with odd wave numbers. This causes the transfer function also to become symmetric around $\tilde{k} = 1/2$ and leads to a filter mask with alternating zeros

$$[h_R, 0, \cdots, h_2, 0, h_1, 0, -h_1, 0, -h_2, \cdots, 0, -h_R]. \tag{13.38}$$

The mask has $4R - 1$ coefficients, $2R - 1$ of which are zero. Figure 13.10 shows the transfer functions optimized with the least squares technique for $R = 2, 3, 4, 5$. The filter with $R = 4$ (a mask with 15 coefficients)

$$\boldsymbol{h} = \{0.6208, 0.1683, 0.0630, 0.0191\}, \tag{13.39}$$

for instance, has an amplitude error of only slightly larger than 1.0 % in the wave number range [0.16, 0.84] and by design no phase error. The convolution with this mask requires 4 multiplications and 7 additions/subtractions.

13.4.3 Analytic Signal‡

A real-valued signal and its Hilbert transform can be combined into a complex-valued signal by

$$g_a = g - \mathrm{i}g_h. \tag{13.40}$$

This complex-valued signal is denoted as the *analytic function* or *analytic signal*. According to Eq. (13.40) the analytic filter has the point spread function

$$a(x) = 1 + \frac{\mathrm{i}}{\pi x} \tag{13.41}$$

and the transfer function

$$\hat{a}(k) = \begin{cases} 2 & k > 0 \\ 1 & k = 0 \\ 0 & k < 0 \end{cases}. \tag{13.42}$$

Thus all negative wave numbers are suppressed. Although the transfer function of the analytic filter is real, it results in a complex signal because it is asymmetric. For a real signal no information is lost by suppressing the negative wave numbers. They can be reconstructed as the Fourier transform of a real signal is Hermitian (Section 2.3.5). The analytic signal can be regarded as just another representation of a real signal with two important properties. The magnitude of the analytic signal gives the *local amplitude*

$$|\mathcal{A}|^2 = \mathcal{I} \cdot \mathcal{I} + \mathcal{H} \cdot \mathcal{H}. \tag{13.43}$$

and the argument the *local phase*

$$\arg(\mathcal{A}) = \arctan\left(\frac{-\mathcal{H}}{\mathcal{I}}\right), \tag{13.44}$$

using $\mathcal{A}$ and $\mathcal{H}$ for the analytic and Hilbert operators, respectively.

The original signal and its Hilbert transform can be obtained from the analytic signal using Eq. (13.40) by

$$\begin{aligned} g(x) &= (g_a(x) + g_a^*(x))/2 \\ g_h(x) &= \mathrm{i}(g_a(x) - g_a^*(x))/2. \end{aligned} \tag{13.45}$$

The concept of the analytic signal also makes it easy to extend the ideas of local phase into multiple dimensions. The transfer function of the analytic operator uses only the positive wave numbers, i. e., only half of the Fourier space. If we extend this partitioning to multiple dimensions, we have more than one choice to partition the Fourier space into two half spaces. Instead of the wave number, we can take the scalar product between the wave number vector $\boldsymbol{k}$ and any unit vector $\bar{\boldsymbol{n}}$ and suppress the half space for which the scalar product $\boldsymbol{k}\bar{\boldsymbol{n}}$ is negative:

$$\hat{a}(\boldsymbol{k}) = \begin{cases} 2 & \boldsymbol{k}\bar{\boldsymbol{n}} > 0 \\ 1 & \boldsymbol{k}\bar{\boldsymbol{n}} = 0 \\ 0 & \boldsymbol{k}\bar{\boldsymbol{n}} < 0 \end{cases}. \tag{13.46}$$

The unit vector $\bar{\boldsymbol{n}}$ gives the direction in which the Hilbert filter is to be applied. The definition Eq. (13.46) of the transfer function of the analytic signal implies that the Hilbert operator can only be applied to directionally filtered signals. This results from the following considerations. For one-dimensional signals we have seen that a discrete Hilbert filter does not work well for small wave numbers (Fig. 13.10). In multiple dimensions this means that a Hilbert filter does not work well if $\tilde{\boldsymbol{k}}\bar{\boldsymbol{n}} \ll 1$. Thus no wave numbers near an orthogonal to the direction of the Hilbert filter may exist, in order to avoid errors.

This fact makes the application of Hilbert filters and thus the determination of the local phase in higher-dimensional signals significantly more complex. It is not sufficient to use bandpass filtered images, e. g., a Laplace pyramid (Section 5.3.3). In addition, the bandpass filtered images must be further decomposed into directional components. At least as many directional components as the dimensionality of the space are required.

13.4.4 Monogenic Signal‡

The extension of the Hilbert transform from a 1-D signal to higher-dimensional signals is not satisfactory because it can only be applied to directionally filtered signals. For wave numbers close to the separation plane, the Hilbert transform does not work. What is really required is an isotropic extension of the Hilbert transform. It is obvious that no scalar-valued transform for a multidimensional signal can be both isotropic and of odd symmetry.

A vector-valued extension of the analytic signal meets both requirements. It is known as the *monogenic signal* and was introduced to image processing by Felsberg and Sommer [38]. The monogenic signal is constructed from the original signal and its *Riesz transform*. The transfer function of the Riesz transform is given by

$$\hat{\boldsymbol{h}}(\boldsymbol{k}) = \mathrm{i}\frac{\boldsymbol{k}}{|\boldsymbol{k}|}. \tag{13.47}$$

The magnitude of the vector $\boldsymbol{h}$ is one for all values of $\boldsymbol{k}$. The Riesz transform is thus isotropic. It is also of odd symmetry because

$$\hat{\boldsymbol{h}}(-\boldsymbol{k}) = -\hat{\boldsymbol{h}}(\boldsymbol{k}). \tag{13.48}$$

The Riesz transform can be applied to a signal of any dimension. For a 1-D signal it reduces to the Hilbert transform.

For a 2-D signal the transfer function of the Riesz transform can be written using polar coordinates as

$$\hat{\boldsymbol{h}}(\boldsymbol{k}) = \mathrm{i}\left[\frac{k\cos\theta}{|k|}, \frac{k\sin\theta}{|k|}\right]^T. \tag{13.49}$$

The convolution mask or PSF of the Riesz transform is given by

$$\boldsymbol{h}(\boldsymbol{x}) = -\frac{\boldsymbol{x}}{2\pi\,|\boldsymbol{x}|^3}. \tag{13.50}$$

The original signal and the signal convolved by the Riesz transform can be combined for a 2-D signal to the 3-D monogenic signal as

$$\boldsymbol{g}_m(\boldsymbol{x}) = [g, h_1 * g, h_2 * g]^T. \tag{13.51}$$

The local amplitude of the monogenic signal is given as the norm of the vector of the monogenic signal as in the case of the analytic signal (Eq. (13.43)):

$$|\boldsymbol{g}_m|^2 = g^2 + (h_1 * g)^2 + (h_2 * g)^2. \tag{13.52}$$

For an intrinsically 1-D signal, it can be proven that the local phase ϕ and the local orientation θ are

$$\tan\phi = \frac{[(h_1 * g)^2 + (h_2 * g)^2]^{1/2}}{g} \tag{13.53}$$

and

$$\tan\theta = \frac{h_2 * g}{h_1 * g}. \tag{13.54}$$

We can thus conclude that the monogenic signal combines an estimate of *local orientation* (Section 13.2) and *local phase* (Section 13.4.1).

13.4.5 Quadrature Filters‡

Quadrature filters are an alternative approach to getting a pair of signals that differ only by a phase shift of 90° ($\pi/2$). It is easiest to introduce the complex form of the quadrature filters. Essentially, the transfer function of a *quadrature filter* is also zero for $\boldsymbol{k}\bar{\boldsymbol{n}} < 0$, like the transfer function of the analytic filter. However, the magnitude of the transfer function is not one but can be any arbitrary real-valued function $h(\boldsymbol{k})$:

$$\hat{q}(\boldsymbol{k}) = \begin{cases} 2h(\boldsymbol{k}) & \boldsymbol{k}\bar{\boldsymbol{n}} > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{13.55}$$

The quadrature filter thus also transforms a real-valued signal into an analytical signal. In contrast to the analytical operator, a wave number weighting is applied. From the complex form of the quadrature filter, we can derive the real quadrature filter pair by observing that they are the part of Eq. (13.55) with even and odd symmetry. Thus

$$\begin{aligned} \hat{g}_+(\boldsymbol{k}) &= (\hat{q}(\boldsymbol{k}) + \hat{q}(-\boldsymbol{k}))/2, \\ \hat{g}_-(\boldsymbol{k}) &= (\hat{q}(\boldsymbol{k}) - \hat{q}(-\boldsymbol{k}))/2. \end{aligned} \tag{13.56}$$

The even and odd part of the quadrature filter pair show a phase shift of 90° and can thus can also be used to compute the local phase.

The best-known quadrature filter pair is the *Gabor filter*. A Gabor filter is a bandpass filter that selects a certain wavelength range around the center wavelength $\boldsymbol{k}_0$ using the Gauss function. The complex transfer function of the Gabor filter is

$$\hat{g}(\boldsymbol{k}) = \begin{cases} \exp\left(|\boldsymbol{k}-\boldsymbol{k}_0|^2\sigma_x^2/2\right) & \boldsymbol{k}\boldsymbol{k}_0 > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{13.57}$$

If $|\boldsymbol{k}_0|\sigma_x > 3$, Eq. (13.57) reduces to

$$\hat{g}(\boldsymbol{k}) = \exp\left(-|\boldsymbol{k}-\boldsymbol{k}_0)|^2\sigma_x^2/2\right). \tag{13.58}$$

Using the relations in Eq. (13.56), the transfer function for the even and odd component are given by

$$\begin{aligned} \hat{g}_+(\boldsymbol{k}) &= \frac{1}{2}\left[\exp\left(-|\boldsymbol{k}-\boldsymbol{k}_0|^2\sigma_x^2/2\right) + \exp\left(-|\boldsymbol{k}+\boldsymbol{k}_0|^2\sigma_x^2/2\right)\right], \\ \hat{g}_-(\boldsymbol{k}) &= \frac{1}{2}\left[\exp\left(-|\boldsymbol{k}-\boldsymbol{k}_0|^2\sigma_x^2/2\right) - \exp\left(-|\boldsymbol{k}+\boldsymbol{k}_0|^2\sigma_x^2/2\right)\right]. \end{aligned} \tag{13.59}$$

The point spread function of these filters can be computed easily with the shift theorem (Theorem 3, p. 52, ≻ R4):

$$\begin{aligned} g_+(\boldsymbol{x}) &= \cos(\boldsymbol{k}_0\boldsymbol{x})\exp\left(-\frac{|\boldsymbol{x}|^2}{2\sigma_x^2}\right), \\ g_-(\boldsymbol{x}) &= \mathrm{i}\sin(\boldsymbol{k}_0\boldsymbol{x})\exp\left(-\frac{|\boldsymbol{x}|^2}{2\sigma_x^2}\right), \end{aligned} \tag{13.60}$$

or combined into a complex filter mask:

$$g(\boldsymbol{x}) = \exp(\mathrm{i}\boldsymbol{k}_0\boldsymbol{x}) \exp\left(-\frac{|\boldsymbol{x}|^2}{2\sigma_x^2}\right). \tag{13.61}$$

Gabor filters are useful for bandpass-filtering images and performing image analysis in the space/wave number domain. Figure 13.11 illustrates an application [*Riemer*, 1991; *Riemer et al.*, 1991]. An image with short wind-generated water surface waves is decomposed by a set of Gabor filters. The center wavelength $\boldsymbol{k}_0$ was set in the x direction, parallel to the wind direction. The filters had the center wavelength in octave distances at 1.2, 2.4, and 4.8 cm wavelengths. The bandwidth was set proportional to the center wave number.

The left column of images in Fig. 13.11 shows the filtering with the even Gabor filter, the right column the local amplitude, which is directly related to the energy of the waves. The filtered images show that waves with different wavelength are partly coupled. In areas where the larger waves have large amplitudes, also the small-scale waves (capillary waves) have large amplitudes. The energy of waves is not equally distributed over the water surface.

An extension of this analysis to image sequences gives a direct insight into the nonlinear wave-wave interaction processes. Figure 13.12 shows the temporal evolution of one row of images from Fig. 13.11. As we will discuss in detail in Section 14.2.4, the slope of the structures in these space-time images towards the time axis is directly proportional to the speed of the moving objects.

It can be observed nicely that the small waves are modulated by the large waves and that the *group velocity* (speed of the wave energy) of the small waves is slower than the phase speed for the capillary waves.

13.4.6 Local Wave Number Determination‡

In order to determine the local wave number, we just need to compute the first spatial derivative of the phase signal (Section 13.4.1, Eq. (13.32)). This derivative has to be applied in the same direction as the Hilbert or quadrature filter has been applied. The phase is given by either

$$\phi(\boldsymbol{x}) = \arctan\left(\frac{-g_h(\boldsymbol{x})}{g(\boldsymbol{x})}\right) \tag{13.62}$$

or

$$\phi(\boldsymbol{x}) = \arctan\left(\frac{-g_+(\boldsymbol{x})}{g_-(\boldsymbol{x})}\right), \tag{13.63}$$

where g_+ and g_- denote the signals filtered with the even and odd part of the quadrature filter.

Direct computation of the partial derivatives from Eqs. (13.62) and (13.63) is not advisable, however, because of the inherent discontinuities in the phase signal. A phase computed with the inverse tangent restricts the phase to the main interval $[-\pi, \pi[$ and thus leads inevitably to a wrapping of the phase signal from π to $-\pi$ with the corresponding discontinuities.

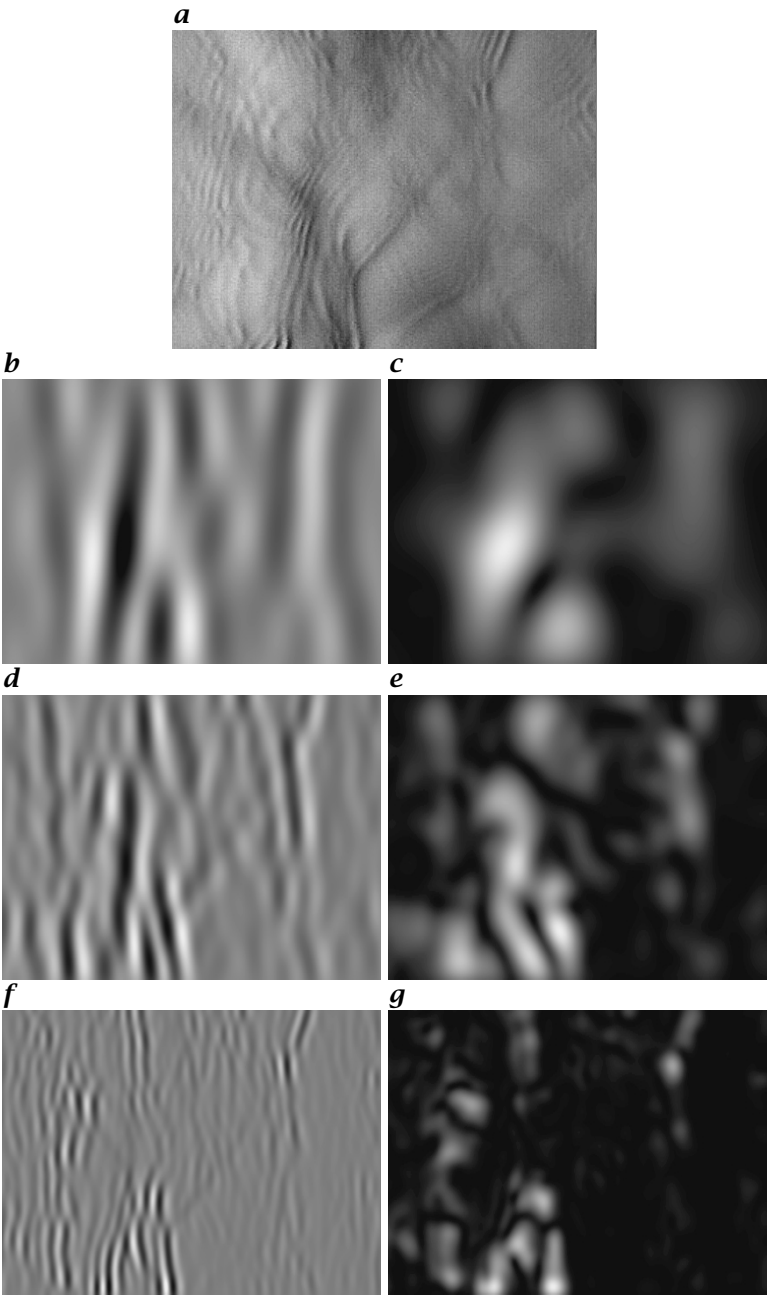

Figure 13.11: *Analysis of an image (**a**, 40 cm × 30 cm) from wind-generated water surface waves. The intensity is proportional to the along-wind component of the slope of the waves. The even part (**b**, **d**, **f**) and squared magnitude (energy, **c**, **e**, **g**) of the Gabor-filtered images with center wavelength at 48, 24, and 12 mm, respectively.*

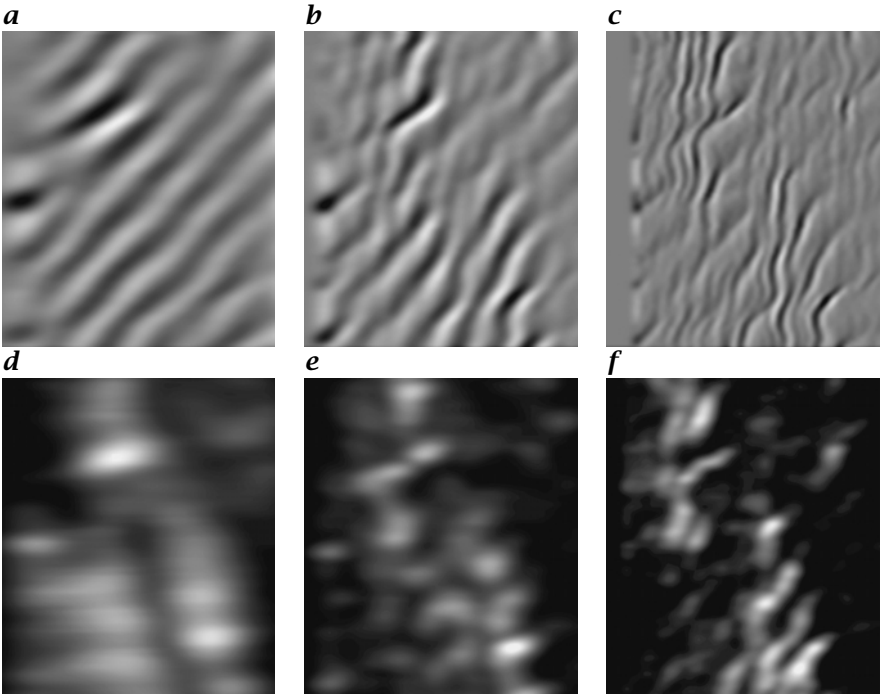

Figure 13.12: *Analysis of a 5 s long space-time slice in wind direction of an image sequence from short wind-generated water surface waves. The time axis is vertically oriented. Even part (**a**–**c**) and squared magnitude (energy, **d**–**f**) of the Gabor-filtered images with center wavelength at 48, 24, and 12 mm, respectively.*

As pointed out by Fleet [42], this problem can be avoided by computing the phase gradient directly from the gradients of $q_+(\boldsymbol{x})$ and $q_-(\boldsymbol{x})$. The result is

$$\begin{aligned} k_p &= \frac{\partial \phi(\boldsymbol{x})}{\partial x_p} \\ &= \frac{\partial}{\partial x_p} \arctan(-q_+(\boldsymbol{x})/q_-(\boldsymbol{x})) \\ &= \frac{1}{q_+^2(\boldsymbol{x}) + q_-^2(\boldsymbol{x})} \left(\frac{\partial q_+(\boldsymbol{x})}{\partial x_p} q_-(\boldsymbol{x}) - \frac{\partial q_-(\boldsymbol{x})}{\partial x_p} q_+(\boldsymbol{x}) \right). \end{aligned} \tag{13.64}$$

This formulation of the phase gradient also eliminates the need for using trigonometric functions to compute the phase signal and is, therefore, significantly faster.

13.5 Tensor Representation by Quadrature Filter Sets‡

13.5.1 Principle‡

Quadrature filters provide another way to analyze simple neighborhoods and to determine both the *local orientation* and the *local wave number*. Historically, this was the first technique for local structure analysis, pioneered by the work of Granlund [56]. The inertia and structure tensor techniques actually appeared later in the literature [8, 90, 146, 147].

The basic idea of the quadrature filter set technique is to extract structures in a certain wave number and direction range. In order to determine local orientation, we must apply a whole set of directional filters, with each filter being sensitive to structures of different orientation. We then compare the filter responses and obtain a maximum filter response from the directional filter whose direction coincides best with that of local orientation. Similarly, the quadrature filter set for different wave number ranges can be set up to determine the local wave number.

If we get a clear maximum in one of the filters but only little response in the others, the local neighborhood contains a locally oriented pattern. If the different filters give comparable responses, the neighborhood contains a distribution of oriented patterns.

So far, the concept seems to be straightforward, but a number of tricky problems needs to be solved. Which properties have to be met by the directional filters in order to ensure an exact determination of local orientation, if at all possible? For computational efficiency, we need to use a minimal number of filters to interpolate the angle of the local orientation. What is this minimal number?

The concepts introduced in this section are based on the work of Granlund [56], Knutsson [98], and Knutsson et al. [100], later summarized in a monograph by Granlund and Knutsson [57]. While the quadrature filter set techniques have been formulated by these authors for multiple dimensions, we will discuss here only the two-dimensional case.

We first discuss the design of quadrature filters that are suitable for the detection of both local orientation and local wave number. This leads to polar separable quadrature filters (Section 13.5.2). In a second step, we show how the orientation vector defined in Section 13.3.3 can be constructed by simple vector addition of the quadrature filter responses (Section 13.5.3). Likewise, in Section 13.5.4 we study the computation of the local wave number. Finally, Section 13.5.5 closes the circle by showing that the structure tensor can also be computed by a set of quadrature filters. Thus the tensor methods discussed in the first part of this chapter (Section 13.3) and the quadrature filter set technique differ only in some subtle points but otherwise give identical results.

13.5.2 Polar Separable Quadrature Filters‡

For an appropriate set of directional filters, each filter should be a rotated copy of the others. This requirement implies that the transfer function of the filters can be separated into an angular part $d(\phi)$ and a wave number part $r(k)$. Such a filter is called *polar separable* and may be conveniently expressed in polar

coordinates

$$\hat{q}(k,\phi) = \hat{r}(k)\hat{d}(\phi), \tag{13.65}$$

where $k = \sqrt{k_1^2 + k_2^2}$ and $\phi = \arctan(k_2/k_1)$ are the magnitude and argument of the wave number, respectively. For a set of directional filters, only the angular part of the transfer function is of importance, while the radial part must be the same for each filter but can be of arbitrary shape. The converse is true for a filter set to determine the local wave number.

Knutsson [98] suggested the following base quadrature filter

$$\begin{aligned} \hat{r}(k) &= \exp\left[-\frac{(\ln k - \ln k_0)^2}{(B/2)^2 \ln 2}\right] \\ \hat{d}(\phi) &= \begin{cases} \cos^{2l}(\phi - \phi_k) & |\phi - \phi_k| < \pi/2 \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{13.66}$$

In this equation, the complex notation for quadrature filters is used (Section 13.4.5). The filter is directed into the angle ϕ_k. The unit vector in this direction is $\bar{\boldsymbol{d}}_k = [\cos\phi_k, \sin\phi_k]$.

The filter is continuous, since the cosine function is zero in the partition plane for the two half spaces ($|\phi - \phi_k| = \pi/2$ or $\bar{\boldsymbol{d}}_k \boldsymbol{k} = 0$). Using the unit vector $\bar{\boldsymbol{d}}_k$ in the direction of the filter, the angular part of the filter can also be written as:

$$\hat{d}(\boldsymbol{k}) = \begin{cases} (\boldsymbol{k}\bar{\boldsymbol{d}}_k)^{2l} & (\boldsymbol{k}\bar{\boldsymbol{d}}_k) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{13.67}$$

The constant k_0 in Eq. (13.66) denotes the peak wave number. The constant B determines the half-width of the wave number in number of octaves and l the angular resolution of the filter. In a logarithmic wave number scale, the filter has the shape of a Gaussian function. Therefore the radial part has a *lognormal* shape.

For the real even and the imaginary odd filter of the quadrature filter pair, the radial part is the same and only the angular part differs:

$$\begin{aligned} \hat{d}_+(\phi) &= \cos^{2l}(\phi - \phi_k) \\ \hat{d}_-(\phi) &= \mathrm{i}\cos^{2l}(\phi - \phi_k)\ \mathrm{sign}(\cos(\phi - \phi_k)). \end{aligned} \tag{13.68}$$

Figure 13.13 shows the radial and angular part of the transfer function for different k_0 and ϕ_k. A set of directional filters is obtained by a suitable choice of different ϕ_k:

$$\phi_k = \frac{\pi k}{K} \quad k = 0, 1, \cdots, K-1. \tag{13.69}$$

Knutsson used four filters with 45° increments in the directions 22.5°, 67.5°, 112.5°, and 157.5°. These directions have the advantage that only *one* filter kernel has to be designed. The kernels for the filter in the other directions are obtained by mirroring the kernels at the axes and diagonals.

These filters were designed in the wave number space. The filter coefficients are obtained by inverse Fourier transformation. If we choose a reasonably small filter mask, we will cut off a number of non-zero filter coefficients. This causes deviations from the ideal transfer function.

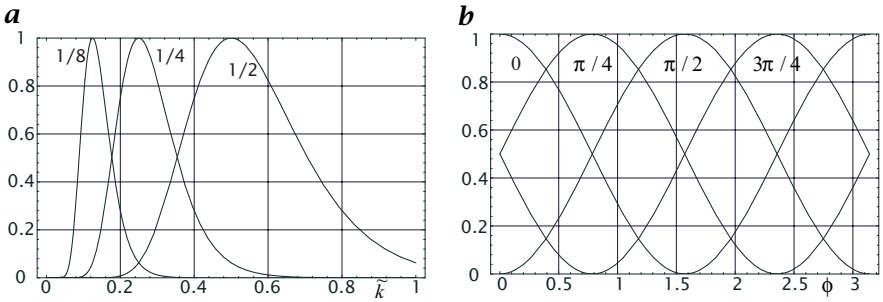

Figure 13.13: ***a*** *Radial and* ***b*** *angular part of quadrature filter according to Eq. (13.66) with* $l = 1$ *and* $B = 2$ *in different directions and with different peak wave numbers.*

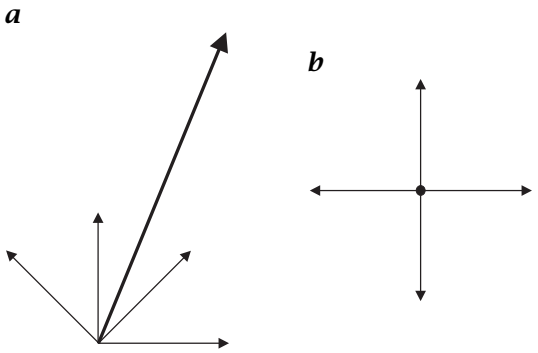

Figure 13.14: *Computation of local orientation by vector addition of the four filter responses. An example is shown where the neighborhood is isotropic concerning orientation: all four filter responses are equal. The angles of the vectors are equal to the filter directions in* ***a*** *and double the filter directions in* ***b*** *.*

Therefore, Knutsson modified the filter kernel coefficient using an optimization procedure in such a way that it approaches the ideal transfer function as closely as possible. It turned out that at least a 15×15 filter mask is necessary to get a good approximation of the anticipated transfer function.

13.5.3 Determination of the Orientation Vector‡

The local orientation can be computed from the responses of the four quadrature filters by vector addition. The idea of the approach is simple. We assign to the individual directional filters an orientation vector. The magnitude of the vector corresponds to the response of the quadrature filter. The direction of the vector is given by the double angle of the filter direction (Section 13.3.3). In this representation each filter response shows how well the orientation of the pattern is directed in the direction of the filter. An estimate of the orientation vector is then given as the vector sum of the filter responses.

Using a represention with complex numbers for the orientation vector, we can write the filter response for the filter in ϕ_k direction as

$$\mathcal{Q}_{\phi_k} = |\mathcal{Q}| \exp(2\mathrm{i}\phi_k). \tag{13.70}$$

Then the orientation vector as the vector sum of the filter responses can be written as

$$\mathcal{O} = \sum_{k=0}^{K-1} \mathcal{Q}_{\phi_k}. \tag{13.71}$$

Figure 13.14 illustrates why an angle doubling is necessary for the vector addition to obtain the orientation vector. An example is taken where the responses from all four filters are equal. In this case the neighborhood contains structures in all directions. Consequently, we observe no local orientation and the vector sum of all filter responses vanishes. This happens if we double the orientation angle (Fig. 13.14b), but not if we omit this step (Fig. 13.14a).

After these more qualitative considerations, we will prove that we can compute the local orientation exactly when the local neighborhood is ideally oriented in an arbitrary direction ϕ_0. As a result, we will also learn the least number of filters we need. We can simplify the computations by only considering the angular terms, as the filter responses show the same wave number dependence. The quick reader can skip this proof.

Using Eq. (13.70), Eq. (13.66), and Eq. (13.69) we can write the angular part of the filter response of the kth filter as

$$\hat{d}_k(\phi_0) = \exp(2\pi\mathrm{i}k/K)\cos^{2l}(\phi_0 - \pi k/K).$$

The cosine function is decomposed into the sum of two complex exponentials:

$$\begin{aligned}
\hat{d}_k(\phi_0) &= \frac{1}{2^{2l}} \exp(2\pi\mathrm{i}k/K)\,[\exp(\mathrm{i}(\phi_0 - \pi k/K)) + \exp(-\mathrm{i}(\phi_0 - \pi k/K))]^{2l} \\
&= \frac{1}{2^{2l}} \exp(2\pi\mathrm{i}k/K) \sum_{j=0}^{2l} \binom{2l}{j} \exp(\mathrm{i}j(\phi_0 - \pi k/K)) \exp(-\mathrm{i}(2l-j)(\phi_0 - \pi k/K)) \\
&= \frac{1}{2^{2l}} \sum_{j=0}^{2l} \binom{2l}{j} \exp(\mathrm{i}(j-l)2\phi_0) \exp(2\pi\mathrm{i}(1+l-j)(k/K)).
\end{aligned}$$

Now we sum up the vectors of all the K directional filters:

$$\sum_{k=0}^{K-1} \hat{d}_k = \frac{1}{2^{2l}} \sum_{j=0}^{2l} \binom{2l}{j} \exp(\mathrm{i}(j-l)2\phi_0) \sum_{k=0}^{K-1} \exp(2\pi\mathrm{i}(1+l-j)(k/K)).$$

The complex double sum can be solved if we carefully analyze the inner sum over k. If $j = l+1$ the exponent is zero. Consequently, the sum is K. Otherwise, the sum represents a geometric series with the factor $\exp(2\pi\mathrm{i}(1+l-j)(k/K))$ and the sum

$$\sum_{k=0}^{K-1} \exp(2\pi\mathrm{i}(1+l-j)(k/K)) = \frac{1 - \exp(2\pi\mathrm{i}(1+l-j))}{1 - \exp(2\pi\mathrm{i}(1+l-j)/K)}. \tag{13.72}$$

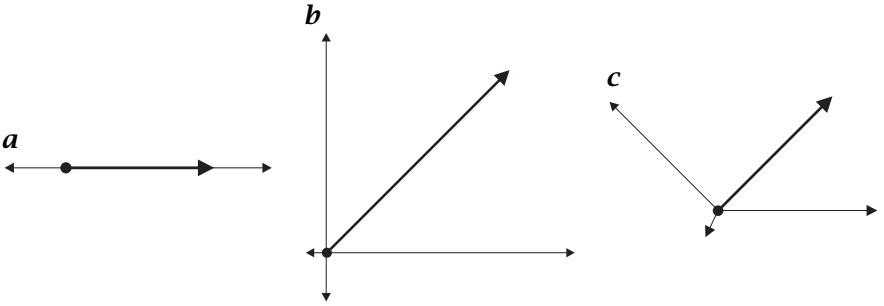

Figure 13.15: *Vector addition of the filter responses from K directional filters to determine local orientation;* ***a*** *$K = 2$;* ***b*** *$K = 3$;* ***c*** *$K = 4$; sum vector shown thicker.*

We can use Eq. (13.72) only if the denominator $\neq 0$ $\forall j = 0, 1, \cdots, 2l$; consequently $K > 1 + l$. With this condition the sum vanishes. This result has a simple geometric interpretation. The sum consists of vectors which are equally distributed on the unit circle. The angle between two consecutive vectors is $2\pi k/K$.

In conclusion, the inner sum in Eq. (13.72) reduces to K for $j = l + 1$, otherwise it is zero. Therefore the sum over j contains only the term with $j = l + 1$. The final result

$$\sum_{k=0}^{K-1} \hat{d}_k = \frac{K}{2^{2l}} \binom{2l}{l+1} \exp(\mathrm{i}2\phi_0) \tag{13.73}$$

shows a vector with the angle of the local orientation doubled. This concludes the proof. ■

The proof of the exactness of the vector addition techniques gives also the minimal number of directional filters required. From $l > 0$ and $K > l + 1$ we conclude that at least $K = 3$ directional filters are necessary. We can also illustrate this condition intuitively. If we have only two filters ($K = 2$), the vector responses of these two filters lie on a line (Fig. 13.15a). Thus orientation determination is not possible. Only with three or four filters can the sum vector point in all directions (Fig. 13.15b, c).

With a similar derivation, we can prove another important property of the directional quadrature filters. The sum over the transfer functions of the K filters results in an isotropic function for $K > l$:

$$\sum_{k=0}^{K-1} \cos^{2l}(\phi - \pi k/K) = \frac{K}{2^{2l}} \binom{2l}{l} \frac{K}{2^{2l}} \frac{(2l)!}{l!^2}. \tag{13.74}$$

In other words, a preferred direction does not exist. The sum of all filter responses gives an *orientation invariant* response. This is also the deeper reason why we can determine local orientation exactly with a very limited number of filters and a simple linear procedure such as vector addition.

13.5.4 Determination of the Local Wave Number‡

The *lognormal* form of the radial part of the quadrature filter sets is the key for a direct estimate of the *local wave number* of a narrowband signal. According to Eq. (13.66), we can write the radial part of the transfer function of the quadrature filter sets as

$$\hat{r}_l(k) = \exp\left[-\frac{(\ln k - \ln k_l)^2}{2\sigma^2 \ln 2}\right]. \tag{13.75}$$

We examine the ratio of the output of two different radial center frequencies k_1 and k_2 and obtain:

$$\begin{aligned}
\frac{\hat{r}_2}{\hat{r}_1} &= \exp\left[-\frac{(\ln k - \ln k_2)^2 - (\ln k - \ln k_1)^2}{2\sigma^2 \ln 2}\right] \\
&= \exp\left[\frac{2(\ln k_2 - \ln k_1)\ln k + \ln^2 k_2 - \ln^2 k_1}{2\sigma^2 \ln 2}\right] \\
&= \exp\left[\frac{(\ln k_2 - \ln k_1)[\ln k - 1/2(\ln k_2 + \ln k_1)]}{\sigma^2 \ln 2}\right] \\
&= \exp\left[\frac{\ln(k/\sqrt{k_2 k_1})\ln(k_2/k_1)}{\sigma^2 \ln 2}\right] \\
&= \left(\frac{k}{\sqrt{k_1 k_2}}\right)^{\ln(k_2/k_1)/(\sigma^2 \ln 2)}
\end{aligned}$$

Generally, the ratio of two different radial filters is directly related to the local wave number. The relation becomes particularly simple if the exponent in the last expression is one. This is the case, for example, if the wave number ratio of the two filters is two ($k_2/k_1 = 2$ and $\sigma = 1$). Then

$$\frac{\hat{r}_2}{\hat{r}_1} = \frac{k}{\sqrt{k_1 k_2}}. \tag{13.76}$$

13.5.5 Determination of the Structure Tensor‡

In this final section we relate the quadrature filter set technique as discussed in Section 13.5 to the tensor technique (Section 13.3). It is shown that the structure tensor can be computed from the responses of these filters. Granlund and Knutsson [57] present the general equation to compute the structure tensor from the quadrature filter responses:

$$\boldsymbol{J}(\boldsymbol{x}) = \sum_{k=0}^{K-1} \mathcal{Q}_k g(\boldsymbol{x}) \left(\alpha \bar{\boldsymbol{d}}_k \otimes \bar{\boldsymbol{d}}_k - \beta \boldsymbol{I}\right), \tag{13.77}$$

where $\mathcal{Q}_k g(\boldsymbol{x})$ is the (amplitude) output of the kth quadrature filter and $\boldsymbol{I}$ the identity matrix. In the two-dimensional case, $\alpha = 4/3$ and $\beta = 1/3$.

We demonstrate this relationship with the quadrature filter set with (the minimum number of) three filters. The three filters point at 0°, 60°, and 120°. Thus

the unit direction vectors are:

$$\begin{aligned} \tilde{\boldsymbol{d}}_0 &= [1,0]^T \\ \tilde{\boldsymbol{d}}_1 &= [1/2,\sqrt{3}/2]^T \\ \tilde{\boldsymbol{d}}_2 &= [-1/2,\sqrt{3}/2]^T \end{aligned} \quad . \tag{13.78}$$

With these values for $\tilde{\boldsymbol{d}}_k$, Eq. (13.77) can be written as

$$\begin{aligned} \boldsymbol{J}(\boldsymbol{x}) &= \mathcal{Q}_0 g(\boldsymbol{x}) \begin{bmatrix} 1 & 0 \\ 0 & -1/3 \end{bmatrix} \\ &+ \mathcal{Q}_1 g(\boldsymbol{x}) \begin{bmatrix} 0 & 1/\sqrt{3} \\ 1/\sqrt{3} & 2/3 \end{bmatrix} \\ &+ \mathcal{Q}_2 g(\boldsymbol{x}) \begin{bmatrix} 0 & -1/\sqrt{3} \\ -1/\sqrt{3} & 2/3 \end{bmatrix}. \end{aligned} \tag{13.79}$$

The matrices give the contribution of the individual quadrature filters to the corresponding elements of the structure tensor. For an isotropically oriented pattern, the output from all quadrature filters is the same. If we set the output to $q(\boldsymbol{x})$, Eq. (13.79) results in the correct structure tensor for an isotropically oriented pattern:

$$\boldsymbol{J}(\boldsymbol{x}) = \begin{bmatrix} q(\boldsymbol{x}) & 0 \\ 0 & q(\boldsymbol{x}) \end{bmatrix}. \tag{13.80}$$

Conversely, for an oriented pattern, the response is $q(\boldsymbol{x})\cos^2(\phi_0 - \phi_k)$ and we obtain

$$\boldsymbol{J}(\boldsymbol{x}) = q(\boldsymbol{x}) \begin{bmatrix} \cos^2(\phi_0) & \sin(2\phi_0)/2 \\ \sin(2\phi_0)/2 & \sin^2(\phi_0) \end{bmatrix}. \tag{13.81}$$

This is the correct form of the structure tensor for an ideally oriented structure in the direction ϕ_0. (This can be shown for instance by checking that the determinant of the matrix is zero and by computing the orientation angle according to Eq. (13.12).)

There is one subtle but important difference between the quadrature filter technique and the structure tensor technique. The quadrature filter technique does not require any averaging to compute the elements of the structure tensor. However, the averaging is an essential element of the direct method. Without averaging, the coherency measure (see Eq. (13.15) in Section 13.3.4) would always be one.

13.6 Further Readings‡

The quadrature filter approach (Section 13.5) is detailed in the monograph of Granlund and Knutsson [57], the inertia tensor method (Section 13.3.7) in a paper by Bigün and Granlund [8]. Poularikas [141] expounds the mathematics of the Hilbert transform. The extension of the analytical signal to higher-dimensional signals (Section 13.4.4) was published only recently by Felsberg and Sommer [38]. More mathematical background to the monogenic signal and geometric algebra for computer vision can be found in the monograph edited by Sommer [176].

14 Motion

14.1 Introduction

Motion analysis long used to be a specialized research area that had not much to do with general image processing. This separation had two reasons. First, the techniques used to analyze motion in image sequences were quite different. Second, the large amount of storage space and computing power required to process image sequences made image sequence analysis available only to a few specialized institutions that could afford to buy the expensive specialized equipment. Both reasons are no longer true. Because of the general progress in image processing, the more advanced methods used in motion analysis no longer differ from those used for other image processing tasks. The rapid progress in computer hardware and algorithms makes the analysis of image sequences now feasible even on standard personal computers and workstations.

Therefore we treat motion in this chapter as just another feature that can be used to identify, characterize, and distinguish objects and to understand scenes. Motion is indeed a powerful feature. We may compare the integration of motion analysis into mainstream image processing with the transition from still photography to motion pictures. Only image sequence analysis allows us to recognize and analyze dynamic processes. Thus far-reaching capabilities become available for scientific and engineering applications including the study of flow; transport; biological growth processes from the molecular to the ecosystem level; diurnal, annual, and interannual variations; industrial processes; traffic; autonomous vehicles and robots — to name just a few application areas. In short, everything that causes temporal changes or makes them visible in our world is a potential subject for image sequence analysis.

The analysis of motion is still a challenging task and requires some special knowledge. Therefore we discuss the basic problems and principles of motion analysis in Section 14.2. Then we turn to the various techniques for motion determination. As in many other areas of image processing, the literature is swamped with a multitude of approaches. This book should not add to the confusion. We emphasize instead the basic principles and we try to present the various concepts in a unified way as filter operations on the space-time images. In this way, the interrelations between the different concepts are made transparent.

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2

Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

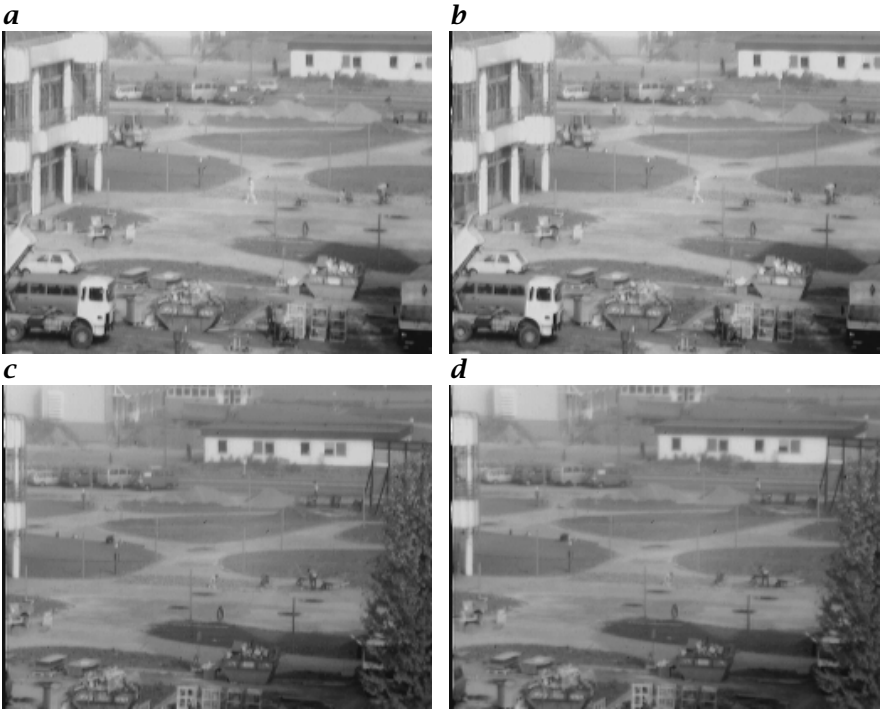

Figure 14.1: a–d *Two pairs of images from the construction area for the new head clinic at Heidelberg University. What has changed from the left to the right images?*

In this sense, we will discuss differential (Section 14.3), tensor (Section 14.4), correlation (Section 14.6), and phase (Section 14.7) techniques as elementary motion estimators.

14.2 Basics

14.2.1 Motion and Gray Value Changes

Intuitively we associate motion with changes. Thus we start our discussion on motion analysis by observing the differences between two images of a sequence. Figure 14.1a and b shows an image pair of a construction area at Heidelberg University. There are differences between the left and right images which are not evident from direct comparison. However, if we subtract one image from the other, the differences immediately become visible (Fig. 14.3a). In the lower left of the image a truck has moved, while the car just behind it is obviously parked. In the center of the image we discover the outline of a pedestrian which is barely visible

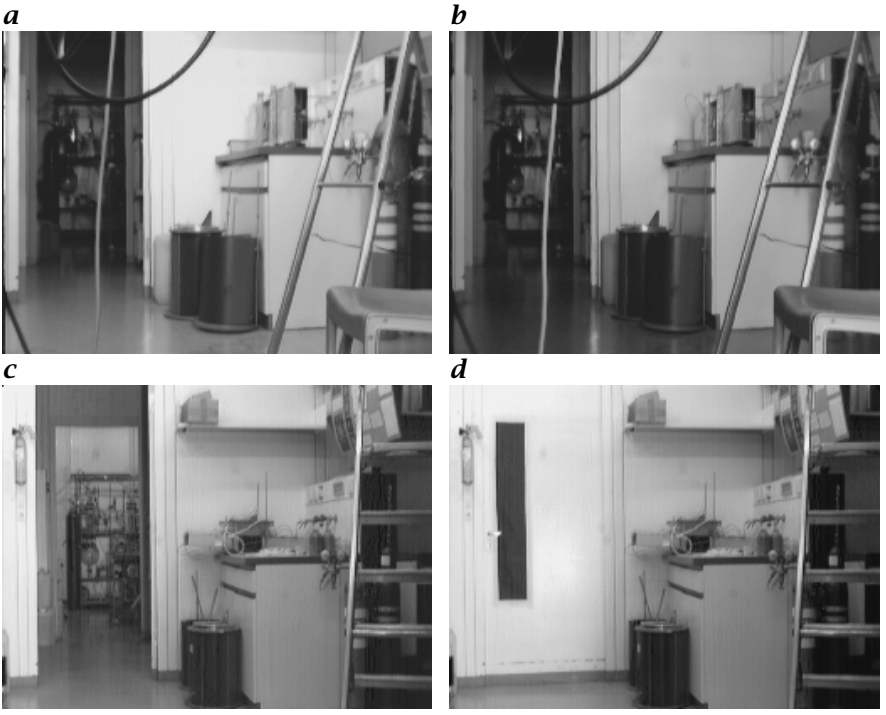

***Figure 14.2: a** to **d** Two pairs of images from an indoor lab scene. What changes can be seen between the left and right images?*

in the original images. The bright spots in a row at the top of the image turn out to be bikers moving along a cycle lane. From the displacement of the double contours we can estimate that they move faster than the pedestrian. Even from this qualitative description, it is obvious that motion analysis helps us considerably in understanding such a scene. It would be much harder to detect the cycle lane without observing the moving bikers.

Figure 14.1c and d show the same scene. Now we might even recognize the change in the original images. If we observe the image edges, we notice that the images have shifted slightly in a horizontal direction. What has happened? Obviously, the camera has been panned. In the difference image Fig. 14.3b all the edges of the objects appear as bright lines. However, the image is dark where the spatial gray value changes are small. Consequently, we can detect motion only in the parts of an image that show gray value changes. This simple observation points out the central role of spatial gray value changes for motion determination.

So far we can sum up our experience with the statement that motion *might* result in temporal gray value changes. Unfortunately, the reverse

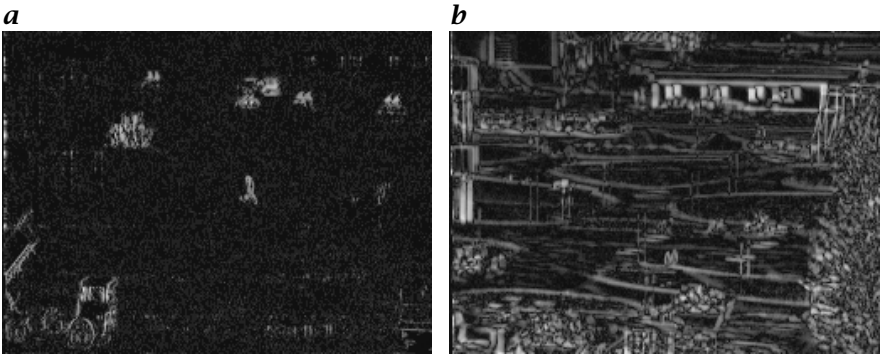

Figure 14.3: *Magnitude of the difference between* ***a*** *images* ***a*** *and* ***b*** *in Fig. 14.1;* ***b*** *images* ***c*** *and* ***d*** *in Fig. 14.1.*

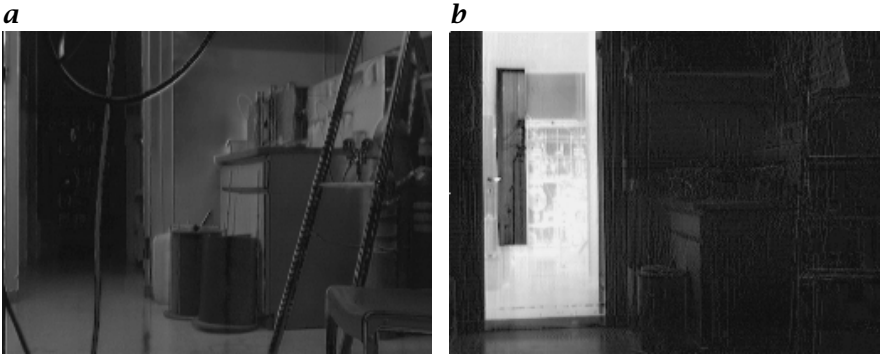

Figure 14.4: *Difference between* ***a*** *images* ***a*** *and* ***b*** *in Fig. 14.2;* ***b*** *images* ***c*** *and* ***d*** *in Fig. 14.2.*

conclusion that all temporal gray value changes are due to motion is not correct. At first glance, the pair of images in Fig. 14.2a and b look identical. Yet, the difference image Fig. 14.4a reveals that some parts in the upper image are brighter than the lower. Obviously the illumination has changed. Actually, a lamp outside the image sector shown was switched off before the image in Fig. 14.2b was taken. Can we infer where this lamp is located? In the difference image we notice that not all surfaces are equally bright. Surfaces which are oriented towards the camera show about the same brightness in both images, while surfaces facing the left hand side are considerably brighter. Therefore we can conclude that the lamp is located to the left outside of the image sector.

Another pair of images (Fig. 14.2c and d) shows a much more complex scene, although we did not change the illumination. We just closed the door of the lab. Of course, we see strong gray value differences where the door is located. The gray value changes, however, extend to the

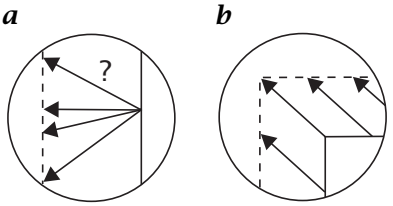

Figure 14.5: *Illustration of the aperture problem in motion analysis:* ***a*** *ambiguity of displacement vectors at an edge;* ***b*** *unambiguity of the displacement vector at a corner.*

floor close to the door and to the objects located to the left of the door (Fig. 14.4b). As we close the door, we also change the illumination in the proximity of the door, especially below the door because less light is reflected into this area.

14.2.2 The Aperture Problem

So far we have learned that estimating motion is closely related to spatial and temporal gray value changes. Both quantities can easily be derived with local operators that compute the spatial and temporal derivatives. Such an operator only "sees" a small sector — equal to the size of its mask — of the observed object. We may illustrate this effect by putting a mask or aperture onto the image.

Figure 14.5a shows an edge that moved from the position of the solid line in the first image to the position of the dotted line in the second image. The motion from image one to two can be described by a *displacement vector*, or briefly, *DV*. In this case, we cannot determine the displacement unambiguously. The displacement vector might connect one point of the edge in the first image with any other point of the edge in the second image (Fig. 14.5a). We can only determine the component of the DV normal to the edge, while the component parallel to the edge remains unknown. This ambiguity is known as the *aperture problem*.

An unambiguous determination of the DV is only possible if a corner of an object is within the mask of our operator (Fig. 14.5b). This emphasizes that we can only gain sparse information on motion from local operators.

14.2.3 The Correspondence Problem

The aperture problem is caused by the fact that we cannot find the corresponding point at an edge in the following image of a sequence, because we have no means of distinguishing the different points at an edge. In this sense, we can comprehend the aperture problem only as a special case of a more general problem, the *correspondence problem*. Generally

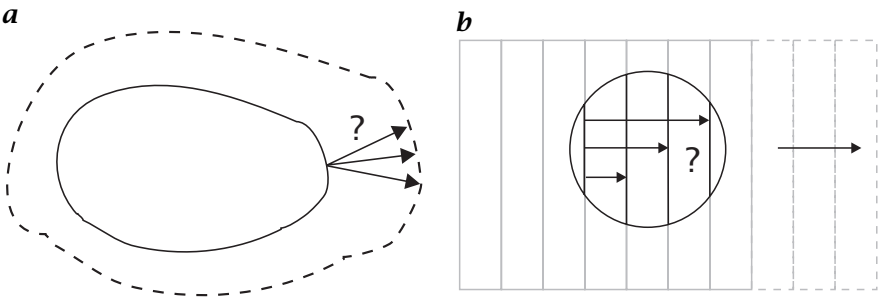

Figure 14.6: *Illustration of the correspondence problem:* ***a*** *deformable two-dimensional object;* ***b*** *regular grid.*

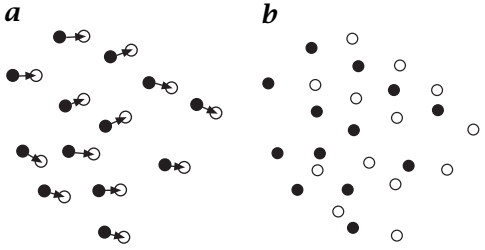

Figure 14.7: *Correspondence problem with indistinguishable particles:* ***a*** *mean particle distance is larger than the mean displacement vector;* ***b*** *the reverse case. Filled and hollow circles: particles in the first and second image.*

speaking, this is that we are unable to find unambiguously corresponding points in two consecutive images of a sequence. In this section we discuss further examples of the correspondence problem.

Figure 14.6a shows a two-dimensional deformable object — like a blob of paint — which spreads gradually. It is immediately obvious that we cannot obtain any unambiguous determination of displacement vectors, even at the edge of the blob. In the inner part of the blob, we cannot make any estimate of the displacements because there are no features visible which we could track.

At first we might assume that the correspondence problem will not occur with rigid objects that show a lot of gray value variations. The grid as an example of a periodic texture, shown in Fig. 14.6b, demonstrates that this is not the case. As long as we observe the displacement of the grid with a local operator, we cannot differentate displacements that differ by multiples of the grid constant. Only when we observe the whole grid does the displacement become unambiguous.

Another variation of the correspondence problem occurs if the image includes many objects of the same shape. One typical case is when small particles are put into a flow field in order to measure the velocity field

(Fig. 14.7). In such a case the particles are indistinguishable and we generally cannot tell which particles correspond to each other. We can find a solution to this problem if we take the consecutive images at such short time intervals that the mean displacement vector is significantly smaller than the mean particle distance. With this additional knowledge, we can search for the nearest neighbor of a particle in the next image. Such an approach, however, will never be free of errors, because the particle distance is statistically distributed.

These simple examples clearly demonstrate the basic problems of motion analysis. On a higher level of abstraction, we can state that the *physical correspondence*, i. e., the real correspondence of the real objects, may not be identical to the *visual correspondence* in the image. The problem has two faces. First, we can find a visual correspondence without the existence of a physical correspondence, as in case of objects or periodic object textures that are indistinguishable. Second, a physical correspondence does not generally imply a visual correspondence. This is the case if the objects show no distinctive marks or if we cannot recognize the visual correspondence because of illumination changes.

14.2.4 Motion as Orientation in Space-Time Images

The discussion in Sections 14.2.1–14.2.3 revealed that the analysis of motion from only two consecutive images is plagued by serious problems. The question arises, whether these problems, or at least some of them, can be overcome if we extend the analysis to more than two consecutive images. With two images, we get just a "snapshot" of the motion field. We do not know how the motion continues in time. We cannot measure accelerations and cannot observe how parts of objects appear or disappear as another object moves in front of them.

In this section, we consider the basics of image sequence analysis in a multidimensional space spanned by one time and one to three space coordinates. Consequently, we speak of a *space-time image*, a *spatiotemporal image*, or simple the $\boldsymbol{x}t$ space.

We can think of a three-dimensional space-time image as a stack of consecutive images which may be represented as an *image cube* as shown in Fig. 14.9. At each visible face of the cube we map a cross section in the corresponding direction. Thus an xt slice is shown on the top face and a yt slice on the right face of the cube. The slices were taken at depths marked by the white lines on the front face, which shows the last image of the sequence.

In a space-time image a pixel extends to a *voxel*, i. e., it represents a gray value in a small volume element with the extensions Δx, Δy, and Δt. Here we confront the limits of our visual imagination when we try to grasp truly 3-D data (compare the discussion in Section 8.1.1).

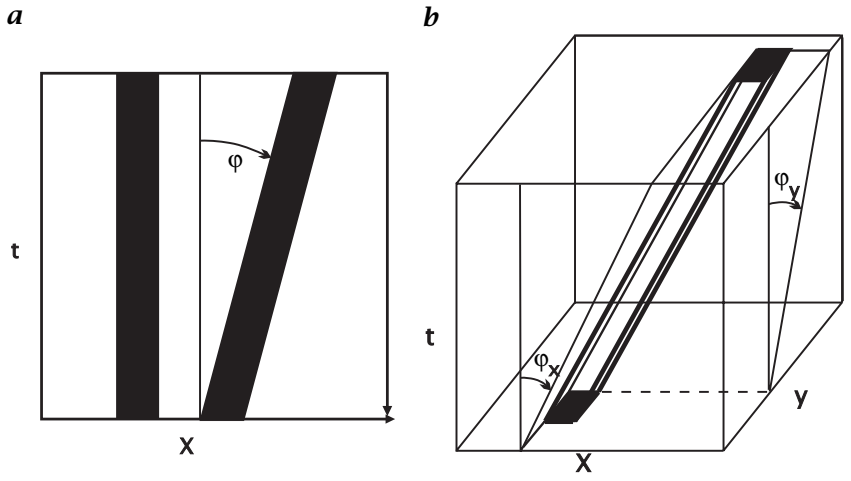

Figure 14.8: *Space-time images:* ***a*** *two-dimensional space-time image with one space and one time coordinate;* ***b*** *three-dimensional space-time image.*

Therefore, we need appropriate representations of such data to make essential features of interest visible.

To analyze motion in space-time images, we first consider a simple example with one space and one time coordinate (Fig. 14.8a). A non-moving 1-D object shows vertically oriented gray value structures. If an object is moving, it is shifted from image to image and thus shows up as an inclined gray value structure. The velocity is directly linked to the orientation in space-time images. In the simple case of a 2-D space-time image, it is given by

$$u = -\tan\varphi, \tag{14.1}$$

where φ is the angle between the t axis and the direction in which the gray values are constant. The minus sign in Eq. (14.1) is because angles are positive counterclockwise. The extension to two spatial dimensions is straightforward and illustrated in Fig. 14.8b:

$$\boldsymbol{u} = -\begin{bmatrix} \tan\varphi_x \\ \tan\varphi_y \end{bmatrix}. \tag{14.2}$$

The angles φ_x and φ_y are defined analogously to the angle between the x and y components of a vector in the direction of the constant gray values and the t axis.

A practical example for this type of analysis is shown in Fig. 14.9. The motion is roughly in the vertical direction, so that the yt cross section can be regarded as a 2-D space-time image. The motion is immediately apparent. When the cars stop at the traffic light, the lines are horizontally

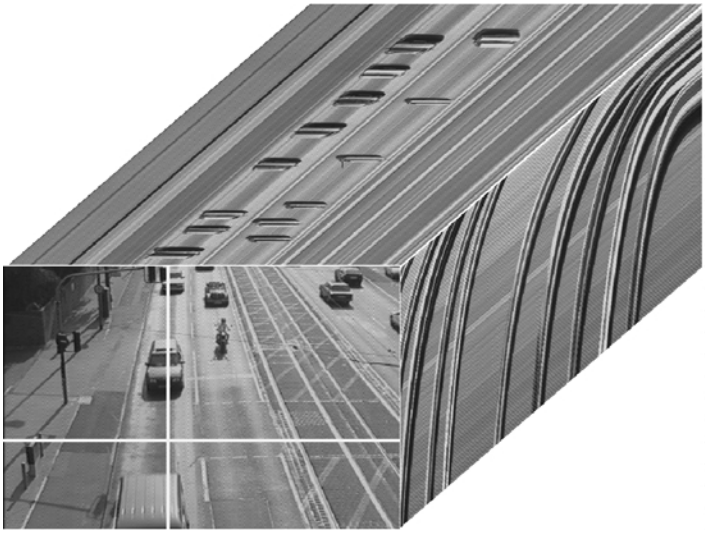

Figure 14.9: *A 3-D image sequence demonstrated with a traffic scene in the Hanauer Landstraße, Frankfurt/Main represented as an image cuboid. The time axis runs into the depth, pointing towards the viewer. On the right side of the cube a yt slice marked by the vertical white line in the xy image is shown, while the top face shows an xt slice marked by the horizontal line (from Jähne [80]).*

oriented, and phases with accelerated and constant speed can easily be recognized.

In summary, we come to the important conclusion that motion appears as *orientation* in space-time images. This fundamental fact forms the basis for motion analysis in $\boldsymbol{xt}$ space. The basic conceptual difference to approaches using two consecutive images is that the velocity is estimated *directly as orientation* in continuous space-time images and not as a discrete *displacement*.

These two concepts differ more than it appears at first glance. Algorithms for motion estimation can now be formulated in continuous $\boldsymbol{xt}$ space and studied *analytically* before a suitable discretization is applied. In this way, we can clearly distinguish the principal flaws of an approach from errors induced by the discretization.

Using more than two images, a more robust and accurate determination of motion can be expected. This is a crucial issue for scientific applications, as pointed out in Chapter 1.

This approach to motion analysis has much in common with the problem of reconstruction of 3-D images from projections (Section 8.6). Actually, we can envisage a geometrical determination of the velocity by observing the transparent three-dimensional space-time image from different points of view. At the right observation angle, we look along the

edges of the moving object and obtain the velocity from the angle between the observation direction and the time axis.

If we observe only the edge of an object, we cannot find such an observation angle unambiguously. We can change the component of the angle along the edge arbitrarily and still look along the edge. In this way, the *aperture problem* discussed in Section 14.2.2 shows up from a different point of view.

14.2.5 Motion in Fourier Domain

Introducing the space-time domain, we gain the significant advantage that we can analyze motion also in the corresponding Fourier domain, the $\boldsymbol{k}\omega$ space. As an introduction, we consider the example of an image sequence in which all the objects are moving with constant velocity. Such a sequence $g(\boldsymbol{x}, t)$ can be described by

$$g(\boldsymbol{x}, t) = g(\boldsymbol{x} - \boldsymbol{u}t). \tag{14.3}$$

The Fourier transform of this sequence is

$$\hat{g}(\boldsymbol{k}, \omega) = \frac{1}{(2\pi)^3} \int_t \int_x g(\boldsymbol{x} - \boldsymbol{u}t) \exp[-2\pi \mathrm{i}(\boldsymbol{k}\boldsymbol{x} - \omega t)] \mathrm{d}^2x \mathrm{d}t. \tag{14.4}$$

Substituting

$$\boldsymbol{x}' = \boldsymbol{x} - \boldsymbol{u}t,$$

we obtain

$$\hat{g}(\boldsymbol{k}, \omega) = \frac{1}{(2\pi)^3} \int_t \left[\int_{x'} g(\boldsymbol{x}') \exp(-2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}') \right] \exp(-2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{u}t) \exp(2\pi \mathrm{i}\omega t) \mathrm{d}^2x \mathrm{d}t.$$

The inner integral covers the spatial coordinates and results in the spatial Fourier transform $\hat{g}(\boldsymbol{k})$ of the image $g(\boldsymbol{x}')$. The outer integral over the time coordinate reduces to a δ function:

$$\hat{g}(\boldsymbol{k}, \omega) = \hat{g}(\boldsymbol{k}) \delta(\boldsymbol{k}\boldsymbol{u} - \omega). \tag{14.5}$$

This equation states that an object moving with the velocity $\boldsymbol{u}$ occupies only a two-dimensional subspace in the three-dimensional $\boldsymbol{k}\omega$ space. Thus it is a line and a plane, in two and three dimensions, respectively. The equation for the plane is given directly by the argument of the δ function in Eq. (14.5):

$$\omega = \boldsymbol{k}\boldsymbol{u}. \tag{14.6}$$

This plane intersects the $k_1 k_2$ plane normally to the direction of the velocity because in this direction the inner product $\boldsymbol{k}\boldsymbol{u}$ vanishes. The slope of the plane, a two-component vector, yields the velocity

$$\nabla_k \omega = \nabla_k (\boldsymbol{k}\boldsymbol{u}) = \boldsymbol{u}.$$

The index k in the gradient operator denotes that the partial derivations are computed with respect to the components of $\boldsymbol{k}$.

From these considerations, it is obvious — at least in principle — how we can determine the velocity in an image sequence showing a constant velocity. We compute the Fourier transform of the sequence and then determine the slope of the plane on which the spectrum of the sequence is located. We can do this best if the scene contains small-scale structures, i. e., high wave numbers which are distributed in many directions. We cannot determine the slope of the plane unambiguously if the spectrum lies on a line instead of a plane. This is the case when the gray value structure is spatially oriented. From the line in Fourier space we only obtain the component of the plane slope in the direction of the spatial local orientation. In this way, we encounter the *aperture problem* (Section 14.2.2) in the $\boldsymbol{k}\omega$ space.

14.2.6 Optical Flow

The examples discussed in Section 14.2.1 showed that motion and gray value changes are not equivalent. In this section, we want to quantify this relation. In this respect, two terms are of importance: the *motion field* and the *optical flow*. The motion field in an image is the real motion of the object in the 3-D scene projected onto the image plane. It is the quantity we would like to extract from the image sequence. The optical flow is defined as the "flow" of gray values at the image plane. This is what we observe. Optical flow and motion field are only equal if the objects do not change the irradiance on the image plane while moving in a scene. Although it sounds reasonable at first glance, a more thorough analysis shows that it is strictly true only in very restricted cases. Thus the basic question is how significant the deviations are, so that in practice we can still stick with the equivalence of optical flow and motion field.

Two classical examples where the projected motion field and the optical flow are not equal were given by Horn [73]. The first is a spinning sphere with a uniform surface of any kind. Such a sphere may rotate around any axes through its center of gravity without causing an optical flow field. The counterexample is the same sphere at rest illuminated by a moving light source. Now the motion field is zero, but the changes in the gray values due to the moving light source cause a non-zero optical flow field.

At this point it is helpful to clarify the different notations for motion with respect to image sequences, as there is a lot of confusion in the literature and many different terms are used. *Optical flow* or *image flow* means the apparent motion at the image plane based on visual perception and has the dimension of a velocity. We denote the optical flow with $\boldsymbol{f} = [f_1, f_2]^T$. If the optical flow is determined from two consecutive images, it appears as a *displacement vector* (*DV*) from the features in the

first to those in the second image. A dense representation of displacement vectors is known as a *displacement vector field* (*DVF*) $\boldsymbol{s} = [s_1, s_2]^T$. An approximation of the optical flow can be obtained by dividing the DVF by the time interval between the two images. It is important to note that optical flow is a concept inherent to continuous space, while the displacement vector field is its discrete counterpart. The *motion field* $\boldsymbol{u} = [u_1, u_2]^T = [u, v]^T$ at the image plane is the projection of the 3-D physical motion field by the optics onto the image plane.

The concept of optical flow originates from *fluid dynamics*. In case of images, motion causes gray values, i. e., an optical signal, to "flow" over the image plane, just as volume elements flow in liquids and gases. In fluid dynamics the continuity equation plays an important role. It expresses the fact that mass is conserved in a flow. Can we formulate a similar continuity equation for gray values and under which conditions are they conserved?

In fluid dynamics, the *continuity equation* for the density ϱ of the fluid is given by

$$\frac{\partial \varrho}{\partial t} + \nabla(\boldsymbol{u}\varrho) = \frac{\partial \varrho}{\partial t} + \boldsymbol{u}\nabla\varrho + \varrho\nabla\boldsymbol{u} = 0. \tag{14.7}$$

This equation is valid for two and three-dimensional flows. It states the conservation of mass in a fluid in a differential form. The temporal change in the density is balanced by the divergence of the flux density $\boldsymbol{u}\varrho$. By integrating the continuity equation over an arbitrary volume element, we can write the equation in an integral form:

$$\int_V \left(\frac{\partial \varrho}{\partial t} + \nabla(\boldsymbol{u}\varrho)\right) \mathrm{d}V = \frac{\partial}{\partial t}\int_V \varrho \mathrm{d}V + \oint_A \varrho\boldsymbol{u}\,\mathrm{d}\boldsymbol{a} = 0. \tag{14.8}$$

The volume integral has been converted into a surface integral around the volume using the Gauss integral theorem. $\mathrm{d}\boldsymbol{a}$ is a vector normal to a surface element $\mathrm{d}A$. The integral form of the continuity equation clearly states that the temporal change of the mass is caused by the net flux into the volume integrated over the whole surface of the volume.

How can we devise a similar continuity equation for the optical flux $\boldsymbol{f}$ — known as the *brightness change constraint equation* (*BCCE*) or *optical flow constraint* (*OFC*) — in computer vision? The quantity analogous to the density ϱ is the irradiance E or the gray value g. However, we should be careful and examine the terms in Eq. (14.7) more closely. The left divergence term $\boldsymbol{f}\nabla g$ describes the temporal brightness change due to a moving gray value gradient. The second term with the divergence of the velocity field $g\nabla\boldsymbol{f}$ seems questionable. It would cause a temporal change even in a region with a constant irradiance if the divergence of the flow field is unequal to zero. Such a case occurs, for instance, when an object moves away from the camera. The irradiance at the image plane

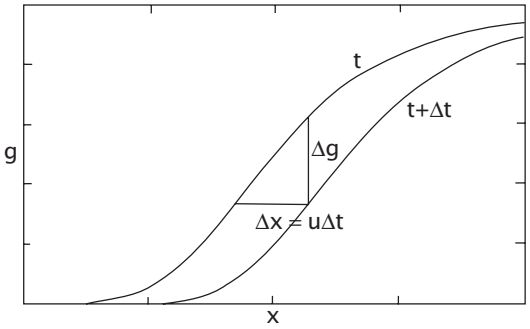

Figure 14.10: *Illustration of the continuity of optical flow in the one-dimensional case.*

remains constant, provided the object irradiance does not change. The collected radiance decreases with the squared distance of the object. However, it is exactly compensated, as also the projected area of the object is decreased by the same factor. Thus we omit the last term in the continuity equation for the optical flux and obtain

$$\frac{\partial g}{\partial t} + \boldsymbol{f}\nabla g = 0. \tag{14.9}$$

In the one-dimensional case, the continuity of the optical flow takes the simple form

$$\frac{\partial g}{\partial t} + f\frac{\partial g}{\partial x} = 0, \tag{14.10}$$

from which we directly get the one-dimensional velocity

$$f = -\frac{\partial g}{\partial t} \Big/ \frac{\partial g}{\partial x}, \tag{14.11}$$

provided that the spatial derivative does not vanish. The velocity is thus given as the ratio of the temporal and spatial derivatives.

This basic relation can also be derived geometrically, as illustrated in Fig. 14.10. In the time interval Δt a gray value is shifted by the distance $\Delta x = u\Delta t$ causing the gray value to change by $g(x, t+\Delta t) - g(x, t)$. The gray value change can also be expressed as the slope of the gray value edge,

$$g(x, t+\Delta t) - g(x, t) = -\frac{\partial g(x, t)}{\partial x}\Delta x = -\frac{\partial g(x, t)}{\partial x}u\Delta t, \tag{14.12}$$

from which, in the limit of $\Delta t \to 0$, the continuity equation for optical flow Eq. (14.10) is obtained.

The continuity or BCCE equation for optical flow at the image plane Eq. (14.9) can in general only be a crude approximation. We have already

touched this subject in the introductory section about motion and gray value changes (Section 14.2.1). This is because of the complex nature of the reflection from opaque surfaces, which depends on the viewing direction, surface normal, and directions of the incident light. Each object receives radiation not only directly from light sources but also from all other objects in the scene that lie in the direct line of sight of the object. Thus the radiant emittance from the surface of one object depends on the position of all the other objects in a scene.

In computer graphics, problems of this type are studied in detail, in search of *photorealistic* computer generated images. A big step towards this goal was a method called *radiosity* which explicitly solved the interrelation of object emittance described above [46]. A general expression for the object emittance — the now famous *rendering equation* — was derived by Kajiya [89].

In image sequence processing, it is in principle required to invert this equation to infer the surface reflectivity from the measured object emittance. The surface reflectivity is a feature invariant to surface orientation and the position of other objects and thus would be ideal for motion estimation. Such an approach is unrealistic, however, because it requires a reconstruction of the 3-D scene before the inversion of the rendering equation can be tackled at all.

As there is no generally valid continuity equation for optical flow, it is important to compare possible additional terms with the terms in the standard BCCE. All other terms basically depend on the rate of changes of a number of quantities but not on the brightness gradients. If the gray value gradient is large, the influence of the additional terms becomes small. Thus we can conclude that the determination of the velocity is most reliable for steep gray value edges while it may be significantly distorted in regions with only small gray value gradients. This conclusion is in agreement with Verri and Poggio [190, 191] findings where they point out the difference between optical flow and the motion field.

Another observation is important. It is certainly true that the historical approach of determining the displacement vectors from only two consecutive images is not robust. In general we cannot distinguish whether a gray value change comes from a displacement or any other source. However, the optical flow becomes more robust in space-time images. We will demonstrate this with two examples.

First, it is possible to separate gray value changes caused by global illumination changes from those caused by motion. Figure 14.11 shows an image sequence of a static scene taken at a rate of 5 frames per minute. The two spatiotemporal time slices (Fig. 14.11a, c), indicated by the two white horizontal lines in Fig. 14.11b, cover a period of about 3.4 h. The upper line covers the high-rise building and the sky. From the sky it can be seen that it was partly cloudy, but sometimes there was direct solar

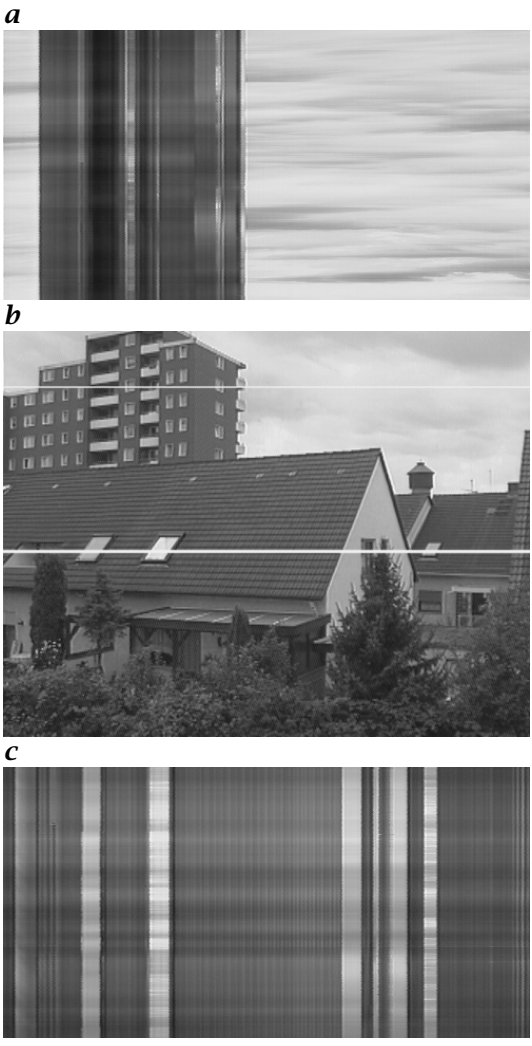

Figure 14.11: *Static scene with illumination changes:* ***a*** xt *cross section at the upper marked row (sky area) in* ***b****;* ***b*** *first image of the sequence;* ***c*** xt *cross section at the lower marked row (roof area) in* ***b****; the time axis spans 3.4 h, running downwards (from Jähne [80]).*

illumination. The lower line crosses several roof windows, walls, and house roofs.

In both slices the illumination changes appear as horizontal stripes which seem to transparently overlay the vertical stripes, indicating a static scene. As a horizontal patterns indicates an object moving with

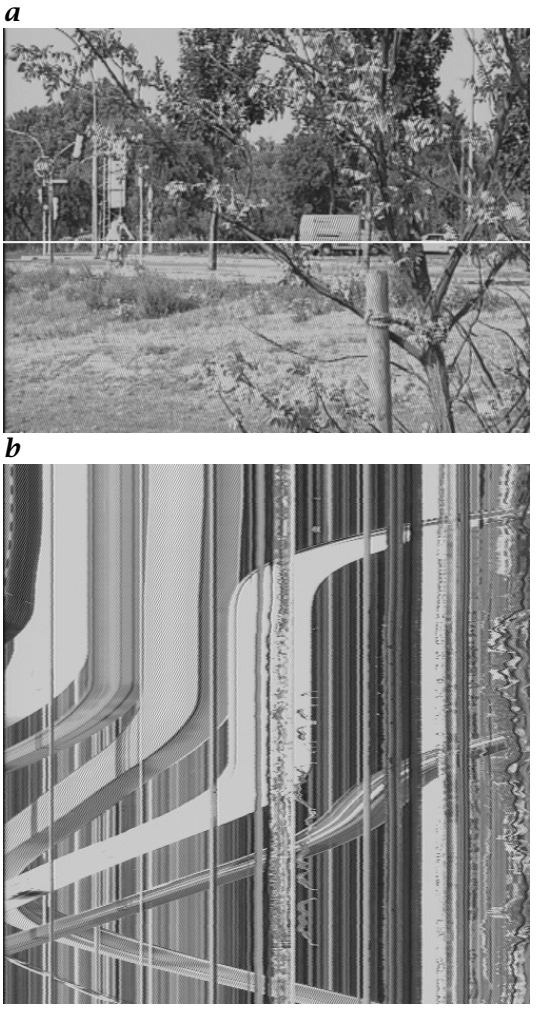

Figure 14.12: *Traffic scene at the border of Hanau, Germany;* ***a*** *last image of the sequence;* ***b*** $x t$ *cross section at the marked line in* ***a****; the time axis spans 20.5 s, running downwards (from Jähne [80]).*

infinite velocity, these patterns can be eliminated, e. g., by directional filtering, without disturbing the motion analysis.

The second example demonstrates that motion determination is still possible in space-time images if occlusions occur and the local illumination of an object is changing because it is turning. Figure 14.12 shows a traffic scene at the city limits of Hanau, Germany. From the last image of the sequence (Fig. 14.12a) we see that a street crossing with a traffic light is observed through the branches of a tree located on the right in

the foreground. One road is running horizontally from left to right, with the traffic light on the left.

The spatiotemporal slice (Fig. 14.12b) has been cut through the image sequence at the horizontal line indicated in Fig. 14.12a. It reveals various occlusions: the car traces disappear under the static vertical patterns of the tree branches and traffic signs. We can also see that the temporal trace of the van shows significant gray value changes because it turned at the street crossing and the illumination conditions are changing while it is moving along in the scene. Nevertheless, the temporal trace is continuous and promises a reliable velocity estimate.

We can conclude that the best approach is to stick to the standard BCCE for motion estimates and use it to develop the motion estimators in this chapter. Because of the wide variety of additional terms this approach still seems to be the most reasonable and most widely applicable, because it contains the fundamental constraint.

14.3 First-Order Differential Methods†

14.3.1 Basics

Differential methods are the classical approach to determine motion from two consecutive images. This chapter discusses the question of how these techniques can be applied to space-time images. The continuity equation for the optical flow (Section 14.2.6), in short the *BCCE* or *OFC*, is the starting point for differential methods:

$$\frac{\partial g}{\partial t} + \boldsymbol{f}\nabla g = 0. \tag{14.13}$$

This single scalar equation contains W unknown vector components in the W-dimensional space. Thus we cannot determine the optical flow $\boldsymbol{f} = [f_1, f_2]^T$ unambiguously. The scalar product $\boldsymbol{f}\nabla g$ is equal to the magnitude of the gray value gradient multiplied by the component of $\boldsymbol{f}$ in the direction of the gradient, i. e., normal to the local gray value edge

$$\boldsymbol{f}\nabla g = f_\perp |\nabla g|.$$

Thus we can only determine the optical flow component normal to the edge. This is the well-known *aperture problem*, which we discussed qualitatively in Section 14.2.2. From Eq. (14.13), we obtain

$$f_\perp = -\frac{\partial g}{\partial t} / |\nabla g| \,. \tag{14.14}$$

Consequently, it is not possible to determine the complete vector with first-order derivatives at a *single* point in the space-time image.

14.3.2 First-Order Least Squares Solution

Instead of a single point, we can use a small neighborhood to determine the optical flow. We assume that the optical flow is constant in this region and discuss in this section under which conditions an unambiguous determination of the optical flow is possible. We still have the two unknowns $\boldsymbol{f} = [f_1, f_2]^T$, but we also have the continuity constraint Eq. (14.13) for the optical flow at many points. Thus we generally end up with an overdetermined equation system. Such a system cannot be solved exactly but only by minimizing an error functional. We seek a solution that minimizes Eq. (14.13) within a local neighborhood in a least squares sense. Thus, the convolution integral

$$\|\boldsymbol{e}\|_2^2 = \int_{-\infty}^{\infty} w(\boldsymbol{x}-\boldsymbol{x}', t-t') \left(f_1 g_x(\boldsymbol{x}') + f_2 g_y(\boldsymbol{x}') + g_t(\boldsymbol{x}')\right)^2 \mathrm{d}^2x'\mathrm{d}t' \tag{14.15}$$

should be minimized. Note that $\boldsymbol{f} = [f_1, f_2]^T$ is constant within the local neighborhood. It depends, of course, as $\|\boldsymbol{e}\|$, on $\boldsymbol{x}$. For the sake of more compact equations, we omit the explicit dependency of g_x, g_y, and g_t on the variable $\boldsymbol{x}'$ in the following equations. The partial derivative $\partial g/\partial p$ is abbreviated by g_p.

In this integral, the square of the residual deviation from the continuity constraint is summed up over a region determined by the size of the window function w. In order to simplify the equations further, we use the following abbreviation for this weighted averaging procedure:

$$\|\boldsymbol{e}\|_2^2 = \overline{\left(f_1 g_x + f_2 g_y + g_t\right)^2}. \tag{14.16}$$

The window function w determines the size of the neighborhood. This makes the least-squares approach very flexible. The averaging in Eq. (14.15) can be but must not be extended in the temporal direction. If we choose a rectangular neighborhood with constant weighting for all points, we end up with a simple *block matching* technique. This corresponds to an averaging with a *box filter*. However, because of the bad averaging properties of box filters (Section 11.3), an averaging with a weighting function that decreases with the distance of the point $[\boldsymbol{x}, t]^T$ from $[\boldsymbol{x}, t]^T$ appears to be a more suitable approach. In continuous space, averaging with a Gaussian filter is a good choice. For discrete images, averaging with a *binomial filter* is most suitable (Section 11.4).

Equation (14.16) can be solved by setting the partial derivatives

$$\begin{aligned} \frac{\partial \|\boldsymbol{e}\|_2^2}{\partial f_1} &= \overline{2 g_x \left(f_1 g_x + f_2 g_y + g_t\right)} \stackrel{!}{=} 0, \\ \frac{\partial \|\boldsymbol{e}\|_2^2}{\partial f_2} &= \overline{2 g_y \left(f_1 g_x + f_2 g_y + g_t\right)} \stackrel{!}{=} 0 \end{aligned} \tag{14.17}$$

to zero. From this condition we obtain the linear equation system

$$\begin{bmatrix} \overline{g_x g_x} & \overline{g_x g_y} \\ \overline{g_x g_y} & \overline{g_y g_y} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = - \begin{bmatrix} \overline{g_x g_t} \\ \overline{g_y g_t} \end{bmatrix}, \tag{14.18}$$

or more compact in matrix notation

$$\boldsymbol{G}\boldsymbol{f} = \boldsymbol{g}. \tag{14.19}$$

The terms $\overline{g_p g_q}$ represent regularized estimates that are composed of convolution and nonlinear point operations. In the operator notation, we can replace it by

$$\mathcal{B}(\mathcal{D}_p \cdot \mathcal{D}_q), \tag{14.20}$$

where $\mathcal{D}_p$ is a suitable discrete first-order derivative operator in the direction p (Chapter 12) and $\mathcal{B}$ an averaging operator (Chapter 11). Thus, the operator expression in Eq. (14.20) includes the following sequence of image processing operators:

1. Apply the convolution operators $\mathcal{D}_p$ and $\mathcal{D}_q$ to the image to obtain images with the first-order derivatives in directions p and q.
2. Multiply the two derivative images pointwise.
3. Convolve the resulting image with the averaging mask $\mathcal{B}$.

Note that the point operation is a nonlinear operation. Therefore, it must not be interchanged with the averaging.

The linear equation system Eq. (14.18) can be solved if the matrix can be inverted. This is the case when the determinant of the matrix is not zero:

$$\det \boldsymbol{G} = \overline{g_x g_x}\,\overline{g_y g_y} - \overline{g_x g_y}^2 \neq 0. \tag{14.21}$$

From this equation, we can deduce two conditions that must be met:

1. Not all partial derivatives g_x and g_y must be zero. In other words, the neighborhood must not consist of an area with constant gray values.
2. The gradients in the neighborhood must not point in the same direction. If this were the case, we could express g_y by g_x except for a constant factor and the determinant of $\boldsymbol{G}$ in Eq. (14.21) would vanish.

The solution for the optical flow $\boldsymbol{f}$ can be written down explicitly because it is easy to invert the 2×2 matix $\boldsymbol{G}$:

$$\boldsymbol{G}^{-1} = \frac{1}{\det \boldsymbol{G}} \begin{bmatrix} \overline{g_y g_y} & -\overline{g_x g_y} \\ -\overline{g_x g_y} & \overline{g_x g_x} \end{bmatrix} \quad \text{if} \quad \det \boldsymbol{G} \neq 0. \tag{14.22}$$

With $\boldsymbol{f} = \boldsymbol{G}^{-1}\boldsymbol{g}$ we then obtain

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = -\frac{1}{\det \boldsymbol{G}} \begin{bmatrix} \overline{g_x g_t}\,\overline{g_y g_y} - \overline{g_y g_t}\,\overline{g_x g_y} \\ \overline{g_y g_t}\,\overline{g_x g_x} - \overline{g_x g_t}\,\overline{g_x g_y} \end{bmatrix}. \tag{14.23}$$

The solution looks still quite complex. It can be simplified considerably by observing that $\boldsymbol{G}$ is a symmetric matrix. Any symmetric matrix can be brought into diagonal form by a rotation of the coordinate system into the so-called *principle-axes* coordinate system. Then the matrix $\boldsymbol{G}'$ reduces to

$$\boldsymbol{G}' = \begin{bmatrix} \overline{g_{x'} g_{x'}} & 0 \\ 0 & \overline{g_{y'} g_{y'}} \end{bmatrix}, \tag{14.24}$$

the determinant $\det \boldsymbol{G}' = \overline{g_{x'} g_{x'}}\ \overline{g_{y'} g_{y'}}$, and the optical flow is

$$\begin{bmatrix} f_{1'} \\ f_{2'} \end{bmatrix} = - \begin{bmatrix} \dfrac{\overline{g_{x'} g_t}}{\overline{g_{x'} g_{x'}}} \\ \dfrac{\overline{g_{y'} g_t}}{\overline{g_{y'} g_{y'}}} \end{bmatrix}. \tag{14.25}$$

This equation reflects in a quantitative way the qualitative discussion about the *aperture problem* discussed in Section 14.2.2. The principal axes are oriented along the directions of the maximum and minimum mean square spatial gray value changes, which are perpendicular to each other. Because the matrix $\boldsymbol{G}$ is diagonal, both changes are uncorrelated. Now, we can distinguish three cases:

1. $\overline{g_{x'} g_{x'}} > 0$, $\overline{g_{y'} g_{y'}} > 0$: spatial gray value changes in all directions. Then both components of the optical flow can be determined.
2. $\overline{g_{x'} g_{x'}} > 0$, $\overline{g_{y'} g_{y'}} = 0$: spatial gray value changes only in x' direction (perpendicularly to an edge). Then only the component of the optical flow in x' direction can be determined (aperture problem). The component of the optical flow parallel to the edge remains unknown.
3. $\overline{g_{x'} g_{x'}} = \overline{g_{y'} g_{y'}} = 0$: no spatial gray value changes in both directions. In the case of a constant region, none of the components of the optical flow can be determined at all.

It is important to note that only the matrix $\boldsymbol{G}$ determines the type of solution of the least-squares approach. In this matrix only spatial and no temporal derivatives occur. This means that the spatial derivatives and thus the spatial structure of the image entirely determines whether and how accurately the optical flow can be estimated.

14.3.3 Error Analysis

Noise may introduce a systematic error into the estimate of the optical flow. Here we show how we can analyze the influence of noise on the determination of optical flow in a very general way. We assume that the image signal is composed of a structure moving with a constant velocity $\boldsymbol{u}$ superimposed by zero-mean isotropic noise:

$$g'(\boldsymbol{x}, t) = g(\boldsymbol{x} - \boldsymbol{u}t) + n(\boldsymbol{x}, t). \tag{14.26}$$

This is a very general approach because we do not rely on any specific form of the gray value structure. The expression $g(\boldsymbol{x} - \boldsymbol{u}t)$ just says that an arbitrary spatial structure is moving with a constant velocity $\boldsymbol{u}$. In this way, a function with three parameters $g(x_1, x_2, t)$ is reduced to a function with only two parameters $g(x_1 - u_1 t, x_2 - u_2 t)$. We further assume that the partial derivatives of the noise function are not correlated with themselves or the partial derivatives of the image patterns. Therefore we use the conditions

$$\overline{n} = 0, \quad \overline{n_p n_q} = \sigma_n^2 \delta_{p-q}, \quad \overline{g_p n_q} = 0, \tag{14.27}$$

and the partial derivatives are

$$\nabla g' = \nabla g + \nabla n \quad g'_t = -\boldsymbol{u}\nabla g + \partial_t n_t. \tag{14.28}$$

These conditions result in the optical flow estimate

$$\boldsymbol{f} = \boldsymbol{u}(\overline{\nabla g \nabla g^T} + \overline{\nabla n \nabla n^T})^{-1} \overline{\nabla g \nabla g^T}. \tag{14.29}$$

The key to understanding this matrix equation is to observe that the noise matrix $\overline{\nabla n \nabla n^T}$ is diagonal in any coordinate system, because of the conditions set by Eq. (14.27). Therefore, we can transform the equation into the principal-axes coordinate system in which $\nabla g \nabla g^T$ is diagonal. Then we obtain

$$\boldsymbol{f} = \boldsymbol{u} \begin{bmatrix} \overline{g_{x'}{}^2} + \sigma_n^2 & 0 \\ 0 & \overline{g_{y'}{}^2} + \sigma_n^2 \end{bmatrix}^{-1} \begin{bmatrix} \overline{g_{x'}{}^2} & 0 \\ 0 & \overline{g_{y'}{}^2} \end{bmatrix}.$$

When the variance of the noise is not zero, the inverse of the first matrix always exists and we obtain

$$\boldsymbol{f} = \boldsymbol{u} \begin{bmatrix} \dfrac{\overline{g_{x'}{}^2}}{\overline{g_{x'}{}^2} + \sigma_n^2} & 0 \\ 0 & \dfrac{\overline{g_{y'}{}^2}}{\overline{g_{y'}{}^2} + \sigma_n^2} \end{bmatrix}. \tag{14.30}$$

This equation shows that the estimate of the optical flow is biased towards lower values. If the variance of the noise is about the squared magnitude of the gradient, the estimated values are only about half of the true values. Thus the differential method is an example of a nonrobust technique because it deteriorates in noisy image sequences.

If the noise is negligible, however, the estimate of the optical flow is correct. This result is in contradiction to the widespread claim that differential methods do not deliver accurate results if the spatial gray value structure cannot be adequately approximated by a first-order Taylor series (see, for example, [172]). Kearney et al. [94], for instance, provided

an error analysis of the gradient approach and concluded that it gives erroneous results as soon as second-order spatial derivatives become significant.

These contradictory findings resolve if we analyze the additional errors in the estimation of optical flow that are introduced by an inadequate discretization of the partial derivative operators (see the discussion on optimal derivative filters in Section 12.3). The error in the optical flow estimate is directly related to the error in the direction of discrete gradient operators (compare also the discussion on orientation estimates in Section 13.3.6). Therefore accurate optical flow estimates require carefully optimized derivative operators such as the optimized regularized gradient operators discussed in Section 12.5.5.

14.3.4 Spatiotemporal Energy Models‡

Models using Gabor-like *quadrature filters* (Section 13.4.5) are common in biological vision. They are the basis for so-called *spatiotemporal energy* models [1, 2, 67]. This term can easily be misunderstood. It is not the kinetic energy of the moving objects that is referred to but the energy (squared amplitude) of a signal at the sensor in a certain $\boldsymbol{k}\omega$ interval. Here we want to compare this type of model with the differential method discussed previously.

One of the simplest models for 1-D motion vision uses just three quadrature filters. This set of directional filters detects objects moving to the right or to the left, and those which are not moving. We denote the squared magnitude of these quadrature operators by $\mathcal{R}$, $\mathcal{L}$, and $\mathcal{S}$. Then we can obtain an estimate of the 1-D optical flow by using the operator [1, 2]:

$$\mathcal{U} = \frac{\mathcal{R} - \mathcal{L}}{\mathcal{S}}. \tag{14.31}$$

An interesting interconnection of this approach with the differential method (Section 14.3.2) can be found, so that the differential method can also be understood as an energy extraction method.

We perform this comparison here for the analysis of 1-D motion, i. e., in a 2-D space-time image. In this case, the solution of the differential method can be written in operator notation according to Eq. (14.25) as

$$\mathcal{U}' = -\frac{\mathcal{B}_{xt}\,(\mathcal{D}_t \cdot \mathcal{D}_x)}{\mathcal{B}_{xt}\,(\mathcal{D}_x \cdot \mathcal{D}_x)}. \tag{14.32}$$

We rewrite this equation with a slight modification to smooth the images with the binomial mask $\mathcal{B}_{xt}$ before we apply the derivative operators, i. e., we use a regularized derivative operator (Section 12.5):

$$\mathcal{U}' = -\frac{\mathcal{B}_{xt}\,[(\mathcal{D}_t\mathcal{B}_{xt}) \cdot (\mathcal{D}_x\mathcal{B}_{xt})]}{\mathcal{B}_{xt}\,[(\mathcal{D}_x\mathcal{B}_{xt}) \cdot (\mathcal{D}_x\mathcal{B}_{xt})]}. \tag{14.33}$$

Smoothing with $\mathcal{B}_{xt}$ means a regularization of the derivative operator. The indices xt indicate that the smoothing is performed along both the temporal and spatial axes. Using the operator identity

$$\mathcal{A}\mathcal{B} = \frac{1}{4}\left[(\mathcal{A} + \mathcal{B})^2 - (\mathcal{A} - \mathcal{B})^2\right] \tag{14.34}$$

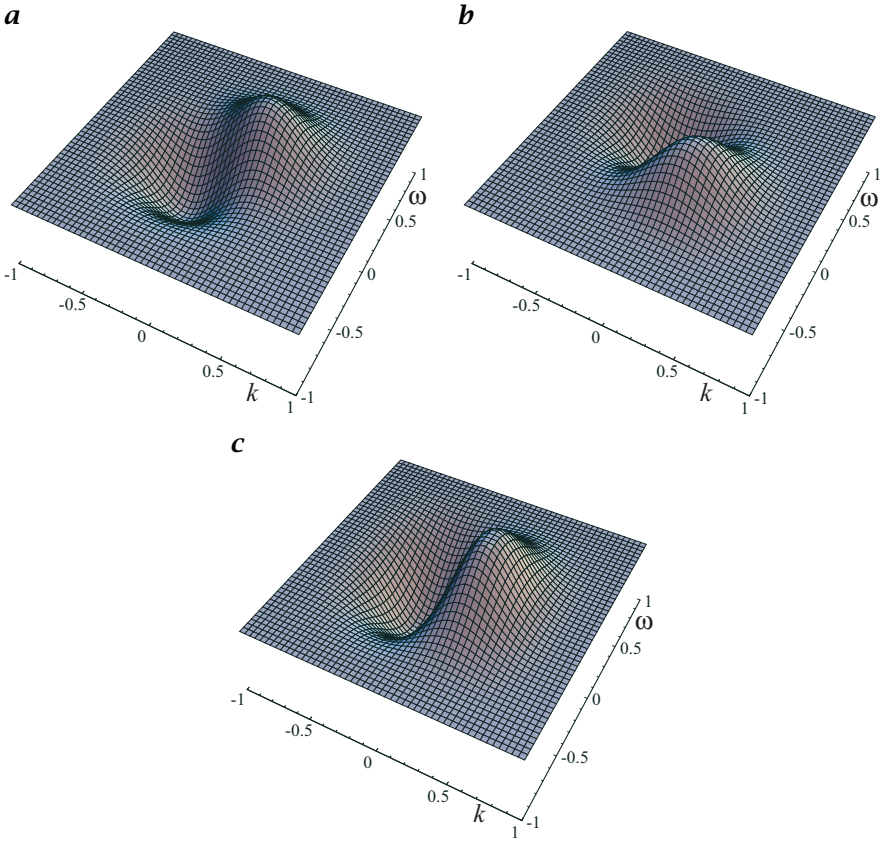

Figure 14.13: *Transfer functions for the convolution operators Eq. (14.35) to detect objects moving right or left, or at rest:* **a** $\mathcal{R}'$, **b** $\mathcal{L}'$, *and* **c** S'.

and the abbreviations

$$\mathcal{R}' = (\mathcal{D}_x + \mathcal{D}_t)\mathcal{B}_{xt}, \quad \mathcal{L}' = (\mathcal{D}_x - \mathcal{D}_t)\mathcal{B}_{xt}, \quad S' = 2\mathcal{D}_x\mathcal{B}_{xt}, \tag{14.35}$$

we can rewrite Eq. (14.33) and obtain a very similar expression as Eq. (14.31):

$$\mathcal{U}' = \frac{\mathcal{B}_{xt}(\mathcal{R}' \cdot \mathcal{R}' - \mathcal{L}' \cdot \mathcal{L}')}{\mathcal{B}_{xt}(S' \cdot S')}. \tag{14.36}$$

The filters $\mathcal{R}'$, $\mathcal{L}'$, and S' are regularized derivative filters. The transfer functions show that objects moving to the right, to the left, and at rest are selected (Fig. 14.13). These filters are *not* quadrature filters. The squaring of the filter responses and further smoothing with $\mathcal{B}_{xt}$, however, approximately results in a phase-independent detection of the squared amplitude as with a quadrature filter under certain conditions. Let us assume a fine-scale periodic structure. The derivative filters will preserve these structures but remove the mean gray value. The subsequent squaring of the zero-mean filter results yields a mean

gray value with half of the squared amplitude of the gray value variations and a rapid spatial oscillation of the gray values with the double wave number (half the wavelength). If the subsequent smoothing removes these fast oscillations, a phase-independent response to the filter is obtained just as with a quadrature filter. In contrast to quadrature filters, this result can only be achieved in regions where the scales of the structures are so fine that the doubled wave number can be removed with the smoothing filter.

14.4 Tensor Methods

The tensor method for the analysis of local orientation has already been discussed in detail in Section 13.3. Since motion constitutes locally oriented structure in space-time images, all we have to do is to extend the tensor method to three dimensions. First, we will revisit the optimization criterion used for the tensor approach in Section 14.4.1 in order to distinguish this technique from the differential method (Section 14.3).

14.4.1 Optimization Strategy

In Section 13.3.1 we stated that the optimum orientation is defined as the orientation that shows the least deviations from the direction of the gradient vectors. We introduced the squared scalar product of the gradient vector and the unit vector representing the local orientation $\bar{\boldsymbol{k}}$ as an adequate measure:

$$(\boldsymbol{\nabla} g \bar{\boldsymbol{k}})^2 = |\boldsymbol{\nabla} g|^2 \cos^2 \left(\sphericalangle(\boldsymbol{\nabla} g, \bar{\boldsymbol{k}}) \right). \tag{14.37}$$

This measure can be used in vector spaces of any dimension. In order to determine orientation in space-time images, we take the spatiotemporal gradient

$$\boldsymbol{\nabla}_{xt} g = \left[\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}, \frac{\partial g}{\partial t} \right]^T = \left[g_x, g_y, g_t \right]^T \tag{14.38}$$

and write

$$(\boldsymbol{\nabla}_{xt} g \bar{\boldsymbol{k}})^2 = |\boldsymbol{\nabla}_{xt} g|^2 \cos^2 \left(\sphericalangle(\boldsymbol{\nabla}_{xt} g, \bar{\boldsymbol{k}}) \right). \tag{14.39}$$

For the 2-D orientation analysis we maximized the expression

$$\int w(\boldsymbol{x} - \boldsymbol{x}') \left(\boldsymbol{\nabla} g(\boldsymbol{x}') \bar{\boldsymbol{k}} \right)^2 \mathrm{d}^W x' = \overline{\left(\boldsymbol{\nabla} g(\boldsymbol{x}') \bar{\boldsymbol{k}} \right)^2} \tag{14.40}$$

in order to find the optimum orientation. For analysis of motion in space-time images, we are not interested in the direction of maximal gray value changes but in the direction of minimal gray value changes. We denote this orientation by the unit vector $\bar{\boldsymbol{e}}_3 = [e_{31}, e_{32}, e_{33}]^T$. This 3-D vector

is, according to the considerations in Section 14.2.4 Eq. (14.2), related to the 2-D velocity vector by

$$\boldsymbol{f} = \frac{1}{e_{33}} \begin{bmatrix} e_{31} \\ e_{32} \end{bmatrix}. \tag{14.41}$$

By analogy to Eq. (14.40) we therefore *minimize*

$$\int w(\boldsymbol{x}-\boldsymbol{x}', t-t') \left(\boldsymbol{\nabla}_{xt} g(\boldsymbol{x}', t')\bar{\boldsymbol{e}}_3\right)^2 \mathrm{d}^2x'\mathrm{d}t' = \overline{\left(\boldsymbol{\nabla}_{xt}\bar{\boldsymbol{e}}\right)^2}. \tag{14.42}$$

The window function w now also extends into the time coordinate and determines the size and shape of the neighborhood around a point $[\boldsymbol{x}, t]^T$ in which the orientation is averaged. Equation (14.42) has to be compared with the corresponding expression that is minimized with the differential method (Eq. (14.16)):

$$\overline{\left(\boldsymbol{f}\boldsymbol{\nabla} g + g_t\right)^2}. \tag{14.43}$$

Note the subtle difference between the two optimization strategies Eqs. (14.42) and (14.43). Both are least squares problems for determining the velocity in such a way that the deviation from the continuity of the optical flow becomes minimal. The two methods differ, however, in the parameters to be estimated. The estimation of a 3-D unit vector turns out to be a so-called *total least squares* problem [76]. This method is more suitable for the problem because all components of the space-time gradient are though to have statistical errors and not only the temporal derivative as in Eq. (14.43).

By analogy to the discussion in Section 13.3.1 we can conclude that the determination of the optical flow in a space-time image is equivalent to finding the *eigenvector* $\bar{\boldsymbol{e}}_3$ of the *smallest* eigenvalue λ_3 of the structure tensor $\boldsymbol{J}$:

$$\boldsymbol{J} = \begin{bmatrix} \overline{g_x g_x} & \overline{g_x g_y} & \overline{g_x g_t} \\ \overline{g_x g_y} & \overline{g_y g_y} & \overline{g_y g_t} \\ \overline{g_x g_t} & \overline{g_y g_t} & \overline{g_t g_t} \end{bmatrix}, \tag{14.44}$$

where $\overline{g_p g_q}$ with $p, q \in \{x, y, t\}$ is given by

$$\overline{g_p g_q}(\boldsymbol{x}, t) = \int w(\boldsymbol{x}-\boldsymbol{x}', t-t') g_p(\boldsymbol{x}', t') g_q(\boldsymbol{x}', t') \mathrm{d}^2x'\mathrm{d}t'. \tag{14.45}$$

At this point, we can compare the tensor method again with the differential technique. While the tensor method essentially performs an eigenvalue analysis of a symmetric tensor with six regularized products of spatial and temporal derivatives, the differential method uses the same products but only five of them. Thus the differential technique misses $\overline{g_t g_t}$. We will see in the next section that this additional term enables the tensor method to detect whether a local neighbor shows a constant velocity or not. This is not possible with the differential method.

14.4.2 Eigenvalue Analysis

Unfortunately, the *eigenvalue analysis* of a symmetric 3×3 tensor is not as simple as for a symmetric 2×2 tensor. In two dimensions we could solve the eigenvalue problem directly. In Section 13.3.3 we transformed the three independent components of the symmetric 2×2 tensor into three parameters: the orientation and the rotation-invariant certainty and coherency measures (Section 13.3.4).

The symmetric 3×3 tensor now contains six independent components and we need to find a corresponding number of parameters that describe the local structure of the space-time image adequately. Again it is useful to decompose these six parameters into rotation-variant and rotation-invariant parameters.

As already mentioned, the solution of the eigenvalue problem cannot be written down readily. It requires a suitable numerical algorithm. We will not detail this problem since it is a nontrivial but standard problem of numerical mathematics for which a number of efficient solutions are available [*Press et al.*, 1992 and *Golub and van Loan*, 1989]. Thus we assume in the following that we have solved the eigenvalue problem and have obtained a set of three orthonormal eigenvectors and three eigenvalues. With the solution of the eigenvalue problem, we have essentially obtained the principal-axes coordinate system in which the structure tensor is diagonal and contains the eigenvalues as diagonal elements:

$$\boldsymbol{J}' = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}. \tag{14.46}$$

Without restricting generality, the eigenvalues are sorted in descending order:

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0. \tag{14.47}$$

The principal-axes coordinate system is spanned by the three eigenvectors. The rotation into this coordinate system requires three independent parameters as we have discussed in Section 7.2.2. Thus, three of the six parameters are used up to describe its orientation in the space-time domain. This information is contained in the three orthonormal eigenvectors.

The remaining parameters are the three rotation-invariant eigenvalues. We will now show how the different classes of local structures in space-time images can be differentiated by the values of the three eigenvalues. This approach will also help us find an efficient implementation of the tensor-based motion analysis. Four different classes of neighborhoods can be distinguished in a space-time image, corresponding to a rank from 0 to 3 for the symmetric tensor:

Constant gray value. All elements and eigenvalues of the structure tensor are zero.

$$\lambda_1 = \lambda_2 = \lambda_3 = 0. \tag{14.48}$$

The rank of the tensor is also zero. Therefore no velocity at all can be obtained. This condition is easy to recognize. The sum of the eigenvalues must be below a critical level, determined by the noise level in the image sequence. As the sum of the eigenvalues is equal to the trace of the tensor, no eigenvalue analysis is required to sort out this condition:

$$\text{trace}(\boldsymbol{J}) = \sum_{p=1}^{3} \overline{g_p g_p} < \gamma, \tag{14.49}$$

where γ is a suitable measure for the noise level in the image sequence. For all points where the condition Eq. (14.49) is met, the eigenvalue analysis can be skipped completely.

Spatial orientation and constant motion. In this case two eigenvalues are zero since the gray values only change in one direction:

$$\lambda_1 > 0 \quad \text{and} \quad \lambda_2 = \lambda_3 = 0. \tag{14.50}$$

The rank of the tensor is one. The spatial gray value structure shows *linear symmetry*. This condition can easily be detected again without performing an eigenvalue analysis, because the determinant of the leading 2×2 subtensor should be below a threshold γ^2:

$$\overline{g_x g_x}\ \overline{g_y g_y} - \overline{g_x g_y}^2 < \gamma^2. \tag{14.51}$$

The eigenvector $\bar{\boldsymbol{e}}_1$ belonging to the only nonzero (i. e., largest) eigenvalue points in the direction of maximum change of gray values. Thus it gives both the spatial orientation and the velocity in this direction. Note that only the *normal velocity*, i. e., the velocity in the direction of the spatial gradient, can be obtained because of the *aperture problem* (Section 14.2.2). The spatial orientation is given by the two spatial coordinates of the eigenvector $\bar{\boldsymbol{e}}_1$ of the largest eigenvalue. As the normal optical flow is in this direction, it is given by

$$\boldsymbol{f}_\perp = -\frac{e_{1t}}{e_{1x}^2 + e_{1y}^2} \begin{bmatrix} e_{1x} \\ e_{1y} \end{bmatrix} \tag{14.52}$$

and the magnitude of the normal optical flow reduces to

$$|\boldsymbol{f}_\perp| = \sqrt{\frac{e_{1t}^2}{e_{1x}^2 + e_{1y}^2}} = \sqrt{\frac{e_{1t}^2}{1 - e_{1t}^2}}. \tag{14.53}$$

Distributed spatial structure and constant motion. In this case only one eigenvalue is zero:

$$\lambda_1, \lambda_2 > 0 \quad \text{and} \quad \lambda_3 = 0. \tag{14.54}$$

As the motion is constant, the principal-axes coordinate system is moving with the scene. The eigenvector $\bar{\boldsymbol{e}}_3$ with the zero eigenvalue points in the direction of constant gray values in the space-time domain. Thus the optical flow is given by

$$\boldsymbol{f} = \frac{1}{e_{3t}} \begin{bmatrix} e_{3x} \\ e_{3y} \end{bmatrix} \tag{14.55}$$

and its magnitude by

$$|\boldsymbol{f}| = \sqrt{\frac{e_{3x}^2 + e_{3y}^2}{e_{3t}^2}} = \sqrt{\frac{1 - e_{3t}^2}{e_{3t}^2}}. \tag{14.56}$$

Distributed spatial structure and non-constant motion. In this case all three eigenvalues are larger than zero and the rank of the tensor is three:

$$\lambda_1, \lambda_2, \lambda_3 > 0. \tag{14.57}$$

No useful optical flow estimate can be obtained in this case.

After this detailed classification, we turn to the question of which three rotation-invariant parameters can be extracted from the structure tensor in order to obtain a useful description of the local structure independent of the velocity and the spatial orientation of the gray scale parameters.

Certainty measure. The first parameter is again a certainty measure that gives a measure for the gray value changes. We have two choices. Either we could take the mean square spatial gradient (trace of the upper 2×2 subtensor) or the mean square spatiotemporal gradient. From a practical point of view the mean square spatial gradient is to be preferred because the spatial gradient does not change in a sequence if the velocity is increasing. The mean square spatiotemporal gradient, however, increases with increasing velocity because higher temporal gradients are added. Thus, surprisingly, the mean square spatial gradient is the better certainty measure:

$$c_c = \overline{g_x g_x} + \overline{g_y g_y}. \tag{14.58}$$

Spatial coherency measure. As a second measure we take the already known coherency measure from the analysis of 2-D local neighborhoods (Section 13.3.4) and denote it here as the spatial coherency measure:

$$c_s = \frac{(\overline{g_x g_x} - \overline{g_y g_y})^2 + 4\overline{g_x g_y}^2}{(\overline{g_x g_x} + \overline{g_y g_y})^2}. \tag{14.59}$$

Its value is between 0 and 1 and decides whether only the normal optical flow or both components of the optical flow can be determined.

Total coherency measure. Finally, we need an additional measure that tells us whether we encounter a local neighborhood with a constant velocity or not. This measure should be independent of the spatial coherency. The following measure using the largest and smallest eigenvalues meets this condition:

$$c_t = \left(\frac{\lambda_1 - \lambda_3}{\lambda_1 + \lambda_3}\right)^2. \tag{14.60}$$

The total coherency measure is one as soon as the eigenvalue λ_3 is zero. The other two eigenvalues may then take any other values. The total coherency approaches zero if all three eigenvalues are equal. In contrast to the other two measures c_c and c_s, the total coherency requires an eigenvalue analysis since the smallest and largest eigenvalues are needed to compute it. There is one caveat with this measure. It is also one with a spatially oriented pattern and a non-constant motion. This special case can be recognized, however, from the condition that both the spatial and total coherency are one but that only one eigenvalue is zero. Another simple criterion is that the eigenvector to the zero eigenvalue lies in the xy plane. This implies that $e_{33} = 0$, so according to Eq. (14.55) we would obtain an infinite value for the optical flow vector.

14.5 Second-Order Differential Methods‡

14.5.1 Basics‡

The first-order differential method has the basic flaw that the continuity of the optical flow gives only one constraint for the two unknown components of the velocity (Section 14.3.1). So far we could only make up for this deficit by modeling the velocity and thus using a whole neighborhood to determine one estimate of the optical flow vector (Section 14.3.2).

An alternative approach is to use multiple feature or multichannel images. Then we can have two or more independent constraints at the same location and thus may be able to determine both components of the optical flow at a single point. The essential point, however, is that the new feature must bring really new information into the image. It does not help at all if a new feature is closely related to those already used.

In this way we come to an important generalization of the differential method. We can apply any preprocessing to image sequences, or can extract arbitrary feature images and apply all the methods discussed so far. If the continuity of the optical flow is preserved for the original image, it is also for any feature image derived from the original image. We can both apply nonlinear point operators as well as any neighborhood operator.

14.5.2 Direct Second-Order Solution‡

We first discuss the technique of Girosi et al. [52]. He applied the continuity of the optical flow to two feature images, namely the horizontal and vertical spatial derivative:

$$\begin{aligned} \boldsymbol{f}\nabla g_x + g_{xt} &= 0 \\ \boldsymbol{f}\nabla g_y + g_{yt} &= 0. \end{aligned} \tag{14.61}$$

The usage of horizontal and vertical derivative images thus results into a second-order differential method with the solution

$$\boldsymbol{f} = -\boldsymbol{H}^{-1}\nabla g_t \quad \text{if} \quad \det \boldsymbol{H} \neq 0, \tag{14.62}$$

where $\boldsymbol{H}$ is the *Hesse matrix* as defined in Eq. (12.4).

14.5.3 Multi-Feature Least Squares Solution‡

If we also include the standard optical flow equation, we end up with an overdetermined linear equation system with three equations and two unknowns:

$$\begin{bmatrix} g_x & g_y \\ g_{xx} & g_{xy} \\ g_{xy} & g_{yy} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = - \begin{bmatrix} g_t \\ g_{xt} \\ g_{yt} \end{bmatrix}. \tag{14.63}$$

In this respect, fusion of images gained from different sensors may be a promising method. Markandey and Flinchbaugh [118], for example, used multispectral imagery, one visible and one IR image. Image sequence processing of scenes illuminated with light sources from different directions has been studied by Woodham [201]. This approach is especially interesting since it has the potential to detect specular reflexes and thus to exclude an important source of errors in motion estimation.

14.5.4 Differential Geometric Modeling‡

The discussion in the last sections clearly showed that the spatial structure of the gray values governs the determination of motion. The first-order differential method does not adequately account for this basic fact as we have just relied on first-order spatial derivatives. Second-order differential methods provide a direct solution, provided the Hesse matrix can be inverted (Eq. (14.61)). In this section, we approach differential methods from a different point of view using *differential geometry*. We assume that the gray value structure in the two consecutive images differs only by a constant displacement $\boldsymbol{s}$:

$$g\left(\boldsymbol{x} - 1/2\boldsymbol{s}, t_1\right) = g\left(\boldsymbol{x} + 1/2\boldsymbol{s}, t_2\right). \tag{14.64}$$

This approach is another formulation of the continuity equation assuming only a translation of the image and neglecting any rotation or deformation of surface elements. We simply assume that the velocity field does not change in a small neighborhood. For the sake of symmetry, we divide the displacement evenly among the two images. With the assumption that the displacement vector $\boldsymbol{s}$

and the size of the surface element are small, we can expand the gray value in both images at the point $\boldsymbol{x} = \boldsymbol{0}$ in a Taylor expansion. First we consider a first-order expansion, i. e., we approximate the gray value distribution by a *plane*:

$$g(\boldsymbol{x} \pm 1/2\boldsymbol{s}) = g_0 + \nabla g \cdot (\boldsymbol{x} \pm 1/2\boldsymbol{s}). \tag{14.65}$$

The planes in both images must be identical except for the displacement $\boldsymbol{s}$. We sort the term in Eq. (14.65) in increasing powers of $\boldsymbol{x}$ in order to be able to perform a coefficient comparison

$$g(\boldsymbol{x} \pm 1/2\boldsymbol{s}) = \underbrace{g_0 \pm 1/2\nabla g\, \boldsymbol{s}}_{\text{offset}} + \underbrace{\nabla g}_{\text{slope}}\, \boldsymbol{x}. \tag{14.66}$$

The first and second term contain the offset and slope of the plane, respectively. We can now estimate the displacement $\boldsymbol{s} = (p, q)^T$ from the condition that both planes must be identical. Consequently, the two coefficients must be identical and we obtain two equations:

$$\begin{aligned} g_0(t_1) - g_0(t_2) &= 1/2\,(\nabla g(t_1) + \nabla g(t_2))\,\boldsymbol{s}, \\ \nabla g(t_1) &= \nabla g(t_2). \end{aligned} \tag{14.67}$$

The second equation states that the gradient must be equal in both images. Otherwise, a plane fit of the spatial gray value does not seem to be a useful representation. The first equation corresponds to the continuity of the optical flow Eq. (14.9). In Eq. (14.67) only the temporal derivative is already expressed in a discrete manner as the difference of the mean gray values in both images. Another refinement is also due to the digitization in time. The gradient is replaced by the mean gradient of both images. Moreover, we use the displacement vector field (DVF) $\boldsymbol{s}$ instead of the optical flow $\boldsymbol{f}$. As expected, a plane fit of the gray value distribution does not yield anything new. We are still only able to estimate the velocity component in the direction of the gray value gradient.

Therefore a Taylor series expansion of Eq. (14.64) to the second order gives

$$\begin{aligned} g(\boldsymbol{x} \pm 1/2\boldsymbol{s}) &= g_0 \\ &+ g_x \cdot (x \pm 1/2s_1) + g_y \cdot (y \pm 1/2s_2) \\ &+ 1/2 g_{xx} \cdot (x \pm 1/2s_1)^2 + 1/2 g_{yy} \cdot (y \pm 1/2s_2)^2 \\ &+ g_{xy} \cdot (x \pm 1/2s_1)\,(y \pm 1/2s_2). \end{aligned}$$

Nagel [128] performed a very similar modeling of the gray value geometry, expanding it in a Taylor series up to second order. However, he ended up with quite complex nonlinear equations, which could be solved easily only for special conditions. He termed them *gray value corner* and *gray value extreme*. The reason for the different results lies in the approach towards the solution. Nagel compares the Taylor expansion in the two images in a least square sense, while here a direct coefficient comparison is performed.

A comparison of the coefficients of the second-order expansion yields six equations in total. The quadratic terms yield three equations which state that all

second-order spatial derivatives must coincide in both images:

$$\begin{aligned} g_{xx}(t_1) &= g_{xx}(t_2), \\ g_{yy}(t_1) &= g_{yy}(t_2), \\ g_{xy}(t_1) &= g_{xy}(t_2). \end{aligned}$$

If this is not the case, either the second-order expansion to the gray value distribution does not adequately fit the gray value distribution or the presumption of a constant displacement in the neighborhood is not valid. The coefficient comparison of the zero- and first-order terms results in the following three equations:

$$\begin{aligned} -(g_0(t_2) - g_0(t_1)) &= \tfrac{1}{2}\left(g_x(t_1) + g_x(t_2)\right) s_1 \\ &\quad + \tfrac{1}{2}\left(g_y(t_1) + g_y(t_2)\right) s_2, \\ -(g_x(t_2) - g_x(t_1)) &= g_{xx} s_1 + g_{xy} s_2, \\ -(g_y(t_2) - g_y(t_1)) &= g_{yy} s_2 + g_{xy} s_1. \end{aligned} \tag{14.69}$$

Surprisingly, the coefficient comparison for the zero-order term (offset) yields the same result as the plane fit Eq. (14.67). This means that the DVF is computed correctly by a simple plane fit, even if the gray value distribution is no longer adequately fitted by a plane but by a second-order polynomial.

The two other equations constitute a simple linear equation system with two unknowns:

$$\begin{bmatrix} g_{xx} & g_{xy} \\ g_{xy} & g_{yy} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = - \begin{bmatrix} g_x(t_2) - g_x(t_1) \\ g_y(t_2) - g_y(t_1) \end{bmatrix}. \tag{14.70}$$

We can easily invert the left 2×2 matrix like we inverted the matrix, provided that $g_{xx}g_{yy} - (g_{xy})^2$ does not vanish. Therefore it is possible to estimate the displacement between two images from a local neighborhood if we take into account the *curvatures* of the gray value distribution. We have not yet discussed the conditions the gray value distribution must meet, for Eq. (14.69) to be invertible. This is the case if either a *gray value extreme* or a *gray value corner* is encountered. As already mentioned, these terms were coined by Nagel [128]. At a gray value extreme (as well as at a saddle point) both principal curvatures are non-zero. Thus Eq. (14.70) can be solved. At a gray value corner, only one principal curvature is zero, but the gradient in this direction is not. Thus the first and second equation from Eq. (14.69) can be used to determine both components of the optical flow vector.

With the differential geometric method no smoothing is required since second-order derivatives at only one point are used. However, for a more robust estimate of derivatives, often regularized derivative operators are used as discussed in Section 12.5. Since convolution operations are commutative, this smoothing could also be applied after computing the derivatives.

The difference in the first-order spatial derivatives between the two images at times t_2 and t_1 in the right-hand vector in Eq. (14.70) is a discrete approximation

of a temporal derivative which can be replaced by a temporal derivative operator. Then, the displacement vector has to be replaced by the optical flow vector. Thus a continuous formulation of the differential geometric method results in

$$\begin{bmatrix} \overline{g_{xx}} & \overline{g_{xy}} \\ \overline{g_{xy}} & \overline{g_{yy}} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = - \begin{bmatrix} \overline{g_{xt}} \\ \overline{g_{yt}} \end{bmatrix}. \tag{14.71}$$

14.6 Correlation Methods

14.6.1 Principle

As with the differential method, the correlation technique is an approach which originates from analyzing the displacement between two consecutive images. To find a characteristic feature from the first image within the second, we take the first image $g(t_1) = g_1$ and compare it with the second image $g(t_2) = g_2$ within a certain search range. Within this range we search for the position of optimum similarity between the two images. When do we regard two features as being similar? The similarity measure should be robust against changes in the illumination. Thus we regard two spatial feature patterns as equal if they differ only by a constant factor α which reflects the difference in illumination. In the language of inner product vector spaces, this means that the two feature vectors g_1 and g_2 are parallel. This can be the case if and only if an equality occurs in the *Cauchy–Schwarz inequality*

$$\left| \int_{-\infty}^{\infty} g_1(\boldsymbol{x}) g_2(\boldsymbol{x} - \boldsymbol{s}) \mathrm{d}^2 x \right|^2 \leq \int_{-\infty}^{\infty} g_1^2(\boldsymbol{x}) \mathrm{d}^2 x \int_{-\infty}^{\infty} g_2^2(\boldsymbol{x} - \boldsymbol{s}) \mathrm{d}^2 x. \tag{14.72}$$

In other words, we need to maximize the *cross-correlation coefficient*

$$r(\boldsymbol{s}) = \frac{\int_{-\infty}^{\infty} g_1(\boldsymbol{x}) g_2(\boldsymbol{x} - \boldsymbol{s}) \mathrm{d}^2 x}{\left(\int_{-\infty}^{\infty} g_1^2(\boldsymbol{x}) \mathrm{d}^2 x \int_{-\infty}^{\infty} g_2^2(\boldsymbol{x} - \boldsymbol{s}) \mathrm{d}^2 x \right)^{1/2}}. \tag{14.73}$$

The cross-correlation coefficient is a useful similarity measure. It is zero for totally dissimilar (orthogonal) patterns, and reaches a maximum of one for similar features.

In a similar way as for the differential method (Section 14.3), the correlation method can be performed by a combination of convolution and point operations. The first step is to introduce a window function w into the definition of the cross-correlation coefficient. This window is

moved around the image to compute the local cross-correlation coefficient. Then Eq. (14.73) becomes

$$r(\boldsymbol{x},\boldsymbol{s}) = \frac{\int_{-\infty}^{\infty} w(\boldsymbol{x}-\boldsymbol{x}')g_1(\boldsymbol{x}')g_2(\boldsymbol{x}'-\boldsymbol{s})\mathrm{d}^2x'}{\left(\int_{-\infty}^{\infty} w(\boldsymbol{x}-\boldsymbol{x}')g_1^2(\boldsymbol{x}')\mathrm{d}^2x' \int_{-\infty}^{\infty} w(\boldsymbol{x}-\boldsymbol{x}')g_2^2(\boldsymbol{x}'-\boldsymbol{s})\mathrm{d}^2x'\right)^{1/2}}. \quad (14.74)$$

The resulting cross-correlation coefficient is a four-dimensional function, depending on the position in the image $\boldsymbol{x}$ and the shift $\boldsymbol{s}$.

14.6.2 Fast Iterative Maximum Search

It is obvious that the correlation method as discussed so far is a very costly operation. A considerable speed-up can be gained if we restrict the computation to a fast approach to search the position of the maximum of r because this is all we are interested in.

One way for a direct computation of the position of the maximum is the approximation of the cross-correlation function in a Taylor series. We expand the cross-correlation coefficient in a second-order Taylor expansion at the position of the maximum $\check{\boldsymbol{s}}$,

$$\begin{aligned} r(\boldsymbol{s}) &\approx r(\check{\boldsymbol{s}}) + \tfrac{1}{2}r_{xx}(\check{\boldsymbol{s}})(s_1-\check{s}_1)^2 + \tfrac{1}{2}r_{yy}(\check{\boldsymbol{s}})(s_2-\check{s}_2)^2 \\ &\quad + r_{xy}(\check{\boldsymbol{s}})(s_1-\check{s}_1)(s_2-\check{s}_2) \\ &= r(\check{\boldsymbol{s}}) + \tfrac{1}{2}(\boldsymbol{s}-\check{\boldsymbol{s}})^T\boldsymbol{H}(\check{\boldsymbol{s}})(\boldsymbol{s}-\check{\boldsymbol{s}}), \end{aligned} \quad (14.75)$$

where $\boldsymbol{H}$ is the Hesse matrix introduced in Eq. (12.4).

We do not know the position of the maximum correlation coefficient. Thus we assume that the second-order derivatives are constant sufficiently close to the position of the maximum and compute it at the position of the previous iteration $\boldsymbol{s}^{(i)}$. If we have no other information, we set the initial estimate to zero: $\boldsymbol{s}^{(0)} = \boldsymbol{0}$. As long as we have not yet found the position of the maximum correlation coefficient, there will be a residual slope at $\boldsymbol{s}^{(i)}$ that can be computed by derivating Eq. (14.75)

$$\nabla r(\boldsymbol{s}^{(i)}) = \boldsymbol{H}(\boldsymbol{s}^{(i)})(\boldsymbol{s}^{(i)}-\check{\boldsymbol{s}}). \quad (14.76)$$

Provided that the Hesse matrix is invertible, we obtain the following iteration

$$\boldsymbol{s}^{(i+1)} = \boldsymbol{s}^{(i)} - \boldsymbol{H}^{-1}(\boldsymbol{s}^{(i)})\nabla r(\boldsymbol{s}^{(i)}) \quad \text{with} \quad \boldsymbol{s}^{(0)} = \boldsymbol{0}. \quad (14.77)$$

This type of iteration is known as Newton-Raphson iteration [143]. In order to estimate the shift, we need to compute only the first- and second-order partial derivatives of the cross-correlation coefficient.

14.6.3 Evaluation and Comparison

In contrast to the differential methods, which are based on the continuity of the optical flux, the correlation approach is insensitive to intensity changes between the two images. This makes correlation-based techniques very useful for stereo-image processing where slight intensity variations always occur between the left and right image because of the two different cameras used. Actually, the fast maximum search described Section 14.6.2 is the standard approach for determining the stereo disparity. Quam [145] used it with a coarse-to-fine control strategy, and Nishihara [132] used it in a modified version, taking the sign of the Laplacian of the Gaussian as a feature. He reports a resolution accuracy of about 0.1 pixel for small displacements. Gelles et al. [51] measured movements in cells with a precision of about 0.02 pixel using the correlation method. However, they used a more costly approach, computing the centroid of a clipped cross-correlation function. The model-adapted approach of Diehl and Burkhardt [31] can be understood as an extended correlation approach as it allows also for rotation and other forms of motion.

The correlation method deviates from all other methods discussed in this work in the respect that it is conceptually based on the comparison of only two images. Even if we extend the correlation technique by multiple correlations to more than two frames, it remains a discrete time-step approach. Thus it lacks the elegance of the other methods, which were formulated in continuous space before being implemented for discrete images. Furthermore, it is obvious that a multiframe extension will be computationally quite expensive.

14.7 Phase Method‡

14.7.1 Principle‡

Except for the costly correlation method, all other methods that compute the optical flow are more or less sensitive to temporal illumination changes. Thus we search for a rich feature which contains the essential information in the images with regard to motion analysis. Fleet and Jepson [45] and Fleet [42] proposed using the phase for the computation of optical flow. We have discussed the crucial role of the phase already in Sections 2.3.6 and 13.4.1. In Section 2.3.6 we demonstrated that the phase of the Fourier transform of a signal carries the essential information. An image can still be recognized when the amplitude information is lost, but not when the phase is lost [112]. Global illumination changes the amplitude of a signal but not its phase.

As an introduction to the phase method, we consider a planar 1-D wave with a wave number k and a circular frequency ω, traveling with a phase speed $u = \omega/k$:

$$g(x,t) = g_0 \exp[-\mathrm{i}(\phi(x,t))] = g_0 \exp[-\mathrm{i}(kx - \omega t)]. \tag{14.78}$$

The position and thus also the displacement is given by the *phase*. The phase depends on both the spatial and temporal coordinates. For a planar wave, the phase varies linearly in time and space,

$$\phi(x,t) = kx - \omega t = kx - ukt, \tag{14.79}$$

where k and ω are the wave number and the frequency of the pattern, respectively. Computing the temporal and spatial derivatives of the phase, i. e., the spatiotemporal gradient, yields both the wave number and the frequency of the moving periodic structure:

$$\nabla_{xt}\phi = \begin{bmatrix} \phi_x \\ \phi_t \end{bmatrix} = \begin{bmatrix} k \\ -\omega \end{bmatrix}. \tag{14.80}$$

Then the velocity is given as the ratio of the frequency to the wave number:

$$u = \frac{\omega}{k} = -\partial_t\phi \,/\partial_x\phi \;. \tag{14.81}$$

This formula is very similar to the estimate based on the optical flow (Eq. (14.11)). In both cases, the velocity is given as a ratio of temporal and spatial derivatives.

Direct computation of the partial derivatives from the phase signal is not advisable because of the inherent discontinuities in the phase signal (restriction to the main interval $[-\pi,\pi[$). As we discussed in Section 13.4.6, it is possible to compute the phase gradients directly from the output of a quadrature filter pair. If we denote the quadrature filter pair with $q_+(x,t)$ and $q_-(x,t)$, the spatiotemporal phase gradient is given by (compare Eq. (13.64)):

$$\nabla_{xt}\phi(x,t) = \frac{q_+(x,t)\,\nabla_{xt}q_-(x,t) - q_-(x,t)\,\nabla_{xt}q_+(x,t)}{q_+^2(x,t) + q_-^2(x,t)}. \tag{14.82}$$

Using Eq. (14.81), the phase derived optical flow f is

$$f = -\frac{q_+\,\partial_t q_- - q_-\,\partial_t q_+}{q_+\,\partial_x q_- - q_-\,\partial_x q_+}. \tag{14.83}$$

14.7.2 Evaluation and Comparison‡

At first sight the phase method appears to offer nothing new. Replacing the gray value by the phase is a significant improvement, however, as the phase is much less dependent on the illumination than the gray value itself. Using only the phase signal, the amplitude of the gray value variations may change without affecting the velocity estimates at all.

So far, we have only considered an ideal periodic gray value structure. Generally, images are composed of gray value structures with different wave numbers. From such a structure we cannot obtain useful phase estimates. Consequently, we need to decompose the image into a set of wave number ranges.

This implies that the phase method is not appropriate to handle two-dimensional shifts. It is essentially a 1-D concept which measures the motion of a linearly

oriented structure, e. g., a planar wave, in the direction of the gray value gradients. From this fact, Fleet and Jepson [44] derived a new paradigm for motion analysis. The image is decomposed with directional filters and in each of the components *normal velocities* are determined.

The 2-D motion field is then composed from these normal velocities. This approach has the advantage that the composition to a complete motion field is postponed to a second processing step which can be adapted to the kind of motion occurring in the images. Therefore this approach can also handle more complex cases such as motion superimposition of transparent objects.

Fleet and Jepson [44] use a set of *Gabor filters* (Section 13.4.5) with an angular resolution of 30° and a bandwidth of 0.8 octaves for the directional decomposition.

Alternatively, a bandpass decomposition and a *Hilbert filter* (Section 13.4.2) can be used. The motivation for this idea is the fact that the decomposition with a set of Gabor filters, as proposed by *Fleet and Jepson*, does not allow easy reconstruction of the original image. The transfer functions of the Gabor filter series do not add up to a unit transfer function but show considerable ripples, as shown by Riemer [154].

A bandpass decomposition, for example using a *Laplacian pyramid* [15, 16], does not share this disadvantage (Section 5.3.3). In addition, it is computationally more efficient. However, we are faced with the problem that no directional decomposition is gained.

Jähne [78, 79] showed how the concept of the Laplacian pyramid can effectively be extended into a *directiopyramidal decomposition*. Each level of the pyramid is further decomposed into two or four directional components which add up directly to the corresponding isotropically filtered pyramid level (see also Section 5.3.4).

14.7.3 From Normal Flow to 2-D Flow‡

As the phase method gives only the normal optical flow, a technique is required to determine the two-dimensional optical flow from the normal flow. The basic relation between the normal and 2-D flow is as follows. We assume that $\boldsymbol{f}_\perp$ is a normal flow vector. It is a result of the projection of the 2-D flow vector $\boldsymbol{f}$ in the direction of the normal flow. Thus we can write:

$$f_\perp = \bar{\boldsymbol{f}}_\perp \boldsymbol{f}, \tag{14.84}$$

where $\bar{\boldsymbol{f}}_\perp$ is a unit vector in the direction of the normal flow. From Eq. (14.84), it is obvious that we can determine the unknown 2-D optical flow in a least squares approach if we have more than two estimates of the normal flow in different directions. In a similar way as in Section 14.3.2, this approach yields the linear equation system

$$\begin{bmatrix} \overline{\bar{f}_{\perp x}\bar{f}_{\perp x}} & \overline{\bar{f}_{\perp x}\bar{f}_{\perp y}} \\ \overline{\bar{f}_{\perp x}\bar{f}_{\perp y}} & \overline{\bar{f}_{\perp y}\bar{f}_{\perp y}} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} \overline{\bar{f}_{\perp x} f_\perp} \\ \overline{\bar{f}_{\perp y} f_\perp} \end{bmatrix} \tag{14.85}$$

with

$$\overline{\bar{f}_{\perp p}\bar{f}_{\perp q}} = \int w(\boldsymbol{x}-\boldsymbol{x}', t-t')\bar{f}_{\perp p}\bar{f}_{\perp q}\mathrm{d}^2x'\mathrm{d}t' \tag{14.86}$$

and

$$\overline{\bar{f}_{\perp p}\bar{f}_{\perp}} = \int w(\boldsymbol{x}-\boldsymbol{x}', t-t')\bar{f}_{\perp p}f_{\perp}\mathrm{d}^2x'\mathrm{d}t'. \tag{14.87}$$

14.8 Further Readings‡

The following monographs on motion analysis are available: Singh [172], Fleet [43], and Jähne [80]. A good survey of motion analysis can also be found in the review articles of Beauchemin and Barron [6] and Jähne and Haußecker [82, Chapter 10]. The latter article also includes the estimation of higher-order motion fields. Readers interested in visual detection of motion in biological systems are referred to the monograph edited by Smith and Snowden [173]. The extension of motion analysis to the estimation of parameters of dynamic processes and illumination variation is described in Haußecker and Fleet [65] and Haußecker [64].

15 Texture

15.1 Introduction

In Chapters 11 and 12 we studied smoothing and edge detection and in Chapter 13 simple neighborhoods. In this chapter, we will take these important building blocks and extend them to analyze complex patterns, known as *texture* in image processing. Actually, textures demonstrate the difference between an artificial world of objects whose surfaces are only characterized by their color and reflectivity properties to that of real-world imagery.

Our visual system is capable of recognizing and distinguishing texture with ease, as can be seen from Fig. 15.1. It appears to be a much more difficult task to characterize and distinguish the rather "diffuse" properties of the texture with some precisely defined parameters that allow a computer to perform this task.

In this chapter we systematically investigate operators to analyze and differentiate between textures. These operators are able to describe even complex patterns with just a few characteristic figures. We thereby reduce the texture recognition problem to the simple task of distinguishing gray values.

How can we define a texture? An arbitrary pattern that extends over a large area in an image is certainly not recognized as a texture. Thus the basic property of a texture is a small elementary pattern which is repeated periodically or quasi-periodically in space like a pattern on a wall paper. Thus, it is sufficient to describe the small elementary pattern and the repetition rules. The latter give the characteristic scale of the texture.

Texture analysis can be compared to the analysis of the structure of solids, a research area studied in solid state physics, chemistry, and mineralogy. A solid state physicist must find out the repetition pattern and the distribution of atoms in the elementary cell. Texture analysis is complicated by the fact that both the patterns and periodic repetition may show significant random fluctuation, as shown in Fig. 15.1.

Textures may be organized in a *hierarchical* manner, i. e., they may look quite different at different scales. A good example is the curtain shown in Fig. 15.1a. On the finest scale our attention is focused on the individual threads (Fig. 15.2a). Then the characteristic scale is the thickness of the threads. They also have a predominant local orientation. On

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2

Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

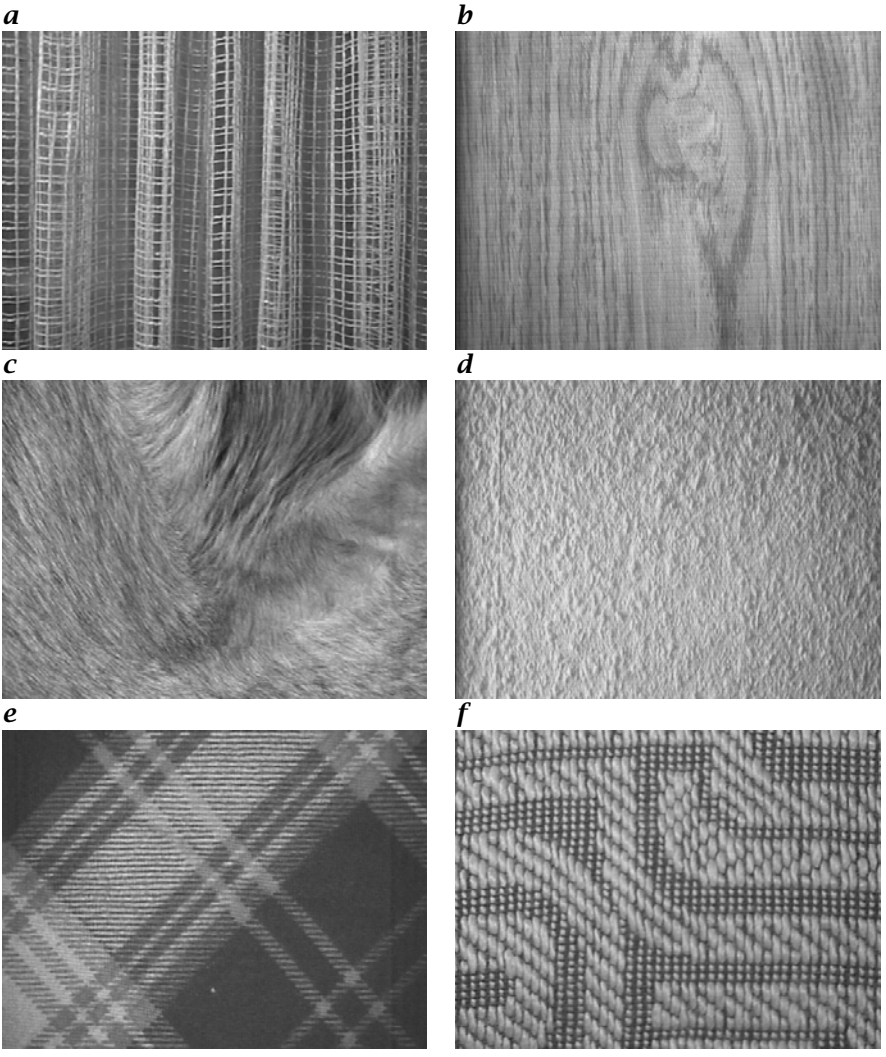

Figure 15.1: *Examples of textures:* ***a*** *curtain;* ***b*** *wood;* ***c*** *dog fur;* ***d*** *woodchip paper;* ***e, f*** *clothes.*

the next coarser level, we will recognize the meshes of the net (Fig. 15.2b). The characteristic scale here shows the size of the meshes. At this level, the local orientation is well distributed. Finally, at an even coarser level, we no longer recognize the individual meshes, but observe the folds of the curtain (Fig. 15.2c). They are characterized by yet another characteristic scale, showing the period of the folds and their orientation. These considerations emphasize the importance of *multiscale texture analysis*.

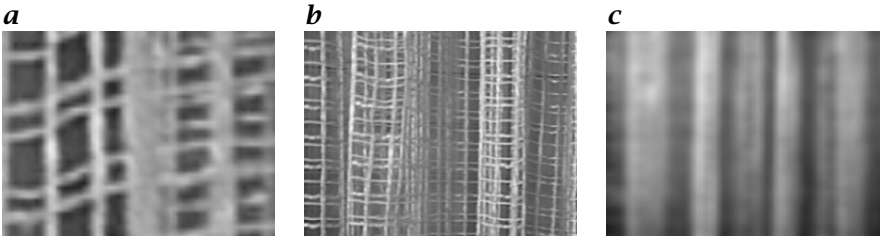

Figure 15.2: *Hierarchical organization of texture demonstrated by showing the image of the curtain in Fig. 15.1a at different resolutions.*

Thus multiscale data structures as discussed in the first part of this book (Chapter 5) are essential for texture analysis.

Generally, two classes of texture parameters are of importance. Texture parameters may or may not be rotation and scale invariant. This classification is motivated by the task we have to perform.

Imagine a typical industrial or scientific application in which we want to recognize objects that are randomly oriented in the image. We are not interested in the orientation of the objects but in their distinction from each other. Therefore, texture parameters that depend on orientation are of no interest. We might still use them but only if the objects have a characteristic shape which then allows us to determine their orientation. We can use similar arguments for scale-invariant features. If the objects of interest are located at different distances from the camera, the texture parameter used to recognize them should also be scale invariant. Otherwise the recognition of the object will depend on distance. However, if the texture changes its characteristics with the scale — as in the example of the curtain in Fig. 15.1a — the scale-invariant texture features may not exist at all. Then the use of textures to characterize objects at different distances becomes a difficult task.

In the examples above, we were interested in the objects themselves but not in their orientation in space. The orientation of surfaces is a key feature for another image processing task, the reconstruction of a three-dimensional scene from a two-dimensional image. If we know the surface of an object shows a uniform texture, we can analyze the orientation and scales of the texture to find the orientation of the surface in space. For this, the characteristic scales and orientations of the texture are needed.

Texture analysis is one of those areas in image processing which still lacks fundamental knowledge. Consequently, the literature contains many different empirical and semiempirical approaches to texture analysis. Here these approaches are not reiterated. In contrast, a rather simple approach to texture analysis is presented which builds complex texture operators from elementary operators.

For texture analysis only four fundamental texture operators are used:

- *mean*,
- *variance*,
- *orientation*,
- *scale*,

which are applied at different levels of the hierarchy of the image processing chain. Once we have, say, computed the local orientation and the local scale, the mean and variance operators can be applied again, now not to the mean and variance of the gray values but to the local orientation and local scale.

These four basic texture operators can be grouped in two classes. The mean and variance are rotation and scale independent, while the orientation and scale operators just determine the orientation and scale, respectively. This important separation between parameters invariant and variant to scale and rotation significantly simplifies texture analysis. The power of this approach lies in the simplicity and orthogonality of the parameter set and the possibility of applying it hierarchically.

15.2 First-Order Statistics

15.2.1 Basics

All texture features based on first-order statistics of the gray value distributions are by definition invariant on any permutation of the pixels. Therefore they do not depend on the orientation of objects and — as long as fine-scale features do not disappear at coarse resolutions — on the scale of the object. Consequently, this class of texture parameter is rotation and scale invariant.

The invariance of first-order statistics to pixel permutations has, however, a significant drawback. Textures with different spatial arrangements but the same gray value distribution cannot be distinguished. Here is a simple example. A texture with equally wide black and white stripes and a texture with a black and white chess board have the same bimodal gray value distribution but a completely different spatial arrangement of the texture.

Thus many textures cannot be distinguished by parameters based on first-order statistics. Other classes of texture parameter are required in addition for a better distinction of different textures.

15.2.2 Local Variance

All parameters that are deviating from the statistics of the gray values of individual pixels are basically independent of the orientation of the

objects. In Section 3.2.2 we learnt to characterize the gray value distribution by the mean, variance, and higher moments. To be suitable for texture analysis, the estimate of these parameters requires to be averaged over a local neighborhood. This leads to a new operator estimating the *local variance*.

In the simplest case, we can select a mask and compute the parameters only from the pixels contained in this window M. The *variance operator*, for example, is then given by

$$v_{mn} = \frac{1}{P-1} \sum_{m',n' \in M} (g_{m-m',n-n'} - \overline{g}_{mn})^2 . \tag{15.1}$$

The sum runs over the P image points of the window. The expression $\overline{g}_{mn}$ denotes the mean of the gray values at the point $[m,n]^T$, computed over the same window M:

$$\overline{g}_{mn} = \frac{1}{P} \sum_{m',n' \in M} g_{m-m',n-n'} . \tag{15.2}$$

It is important to note that the variance operator is nonlinear. However, it resembles the general form of a neighborhood operation — a convolution. Combining Eqs. (15.1) and (15.2), we can show the variance operator is a combination of linear convolution and nonlinear point operations

$$V_{mn} = \frac{1}{P-1} \left[\sum_{m',n' \in M} g^2_{m-m',n-n'} - \left(\frac{1}{P} \sum_{m',n' \in M} g_{m-m',n-n'} \right)^2 \right], \tag{15.3}$$

or, in operator notation,

$$\mathcal{V} = \mathcal{R}(\mathcal{I} \cdot \mathcal{I}) - (\mathcal{R} \cdot \mathcal{R}). \tag{15.4}$$

The operator $\mathcal{R}$ denotes a smoothing over all the image points with a box filter of the size of the window M. The operator $\mathcal{I}$ is the identity operator. Therefore the operator $\mathcal{I} \cdot \mathcal{I}$ performs a nonlinear point operation, namely the squaring of the gray values at each pixel. Finally, the variance operator subtracts the square of a smoothed gray value from the smoothed squared gray values. From discussions on smoothing in Section 11.3 we know that a box filter is not an appropriate smoothing filter. Thus we obtain a better variance operator if we replace the box filter $\mathcal{R}$ with a binomial filter $\mathcal{B}$

$$\mathcal{V} = \mathcal{B}(\mathcal{I} \cdot \mathcal{I}) - (\mathcal{B} \cdot \mathcal{B}). \tag{15.5}$$

We know the variance operator to be isotropic. It is also scale independent if the window is larger than the largest scales in the textures

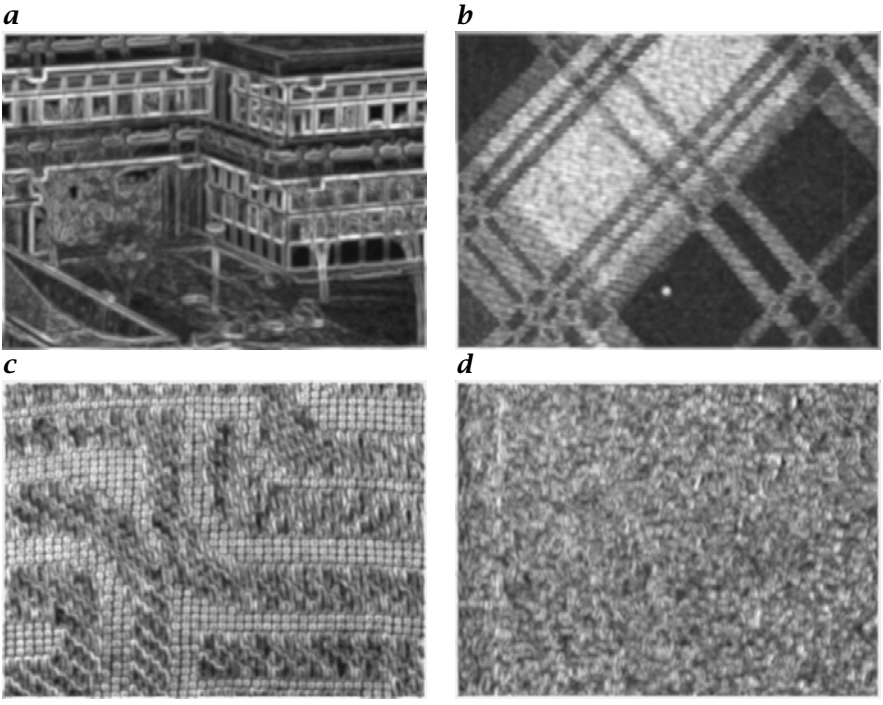

Figure 15.3: *Variance operator applied to different images:* ***a*** *Fig. 11.6a;* ***b*** *Fig. 15.1e;* ***c*** *Fig. 15.1f;* ***d*** *Fig. 15.1d.*

and if no fine scales of the texture disappear because the objects are located further away from the camera. This suggests that a scale-invariant texture operator only exists if the texture itself is scale invariant.

The application of the variance operator Eq. (15.5) with $\mathcal{B}^{16}$ to several images is shown in Fig. 15.3. In Fig. 15.3a, the variance operator turns out to be an isotropic edge detector, because the original image contains areas with more or less uniform gray values.

The other three examples in Fig. 15.3 show variance images from textured surfaces. The variance operator can distinguish the areas with the fine horizontal stripes in Fig. 15.1e from the more uniform surfaces. They appear as uniform bright areas in the variance image (Fig. 15.3b). The variance operator cannot distinguish between the two textures in Fig. 15.3c. As the resolution is still finer than the characteristic repetition scale of the texture, the variance operator does not give a uniform estimate of the variance in the texture. The chipwood paper (Fig. 15.3d) also gives a non-uniform response to the variance operator because the pattern shows significant random fluctuations.

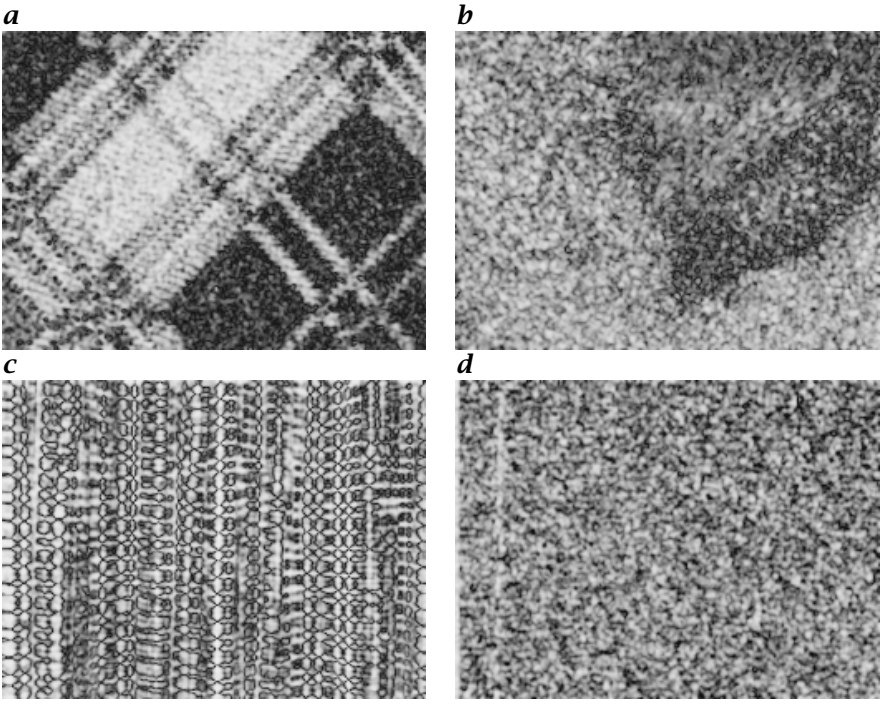

Figure 15.4: *Coherence of local orientation of* **a** *piece of cloth with regions of horizontal stripes (Fig. 15.1e),* **b** *dog fur (Fig. 15.1c),* **c** *curtain (Fig. 15.1a), and* **d** *woodchip wall paper.*

15.2.3 Higher Moments

Besides the variance, we could also use the higher moments of the gray value distribution as defined in Section 3.2.2 for a more detailed description. The significance of this approach may be illustrated with examples of two quite different gray value distributions, a normal and a bimodal distribution:

$$p(g) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{g-\overline{g}}{2\sigma^2}\right), \quad p'(g) = \frac{1}{2}\left(\delta(\overline{g}+\sigma) + \delta(\overline{g}-\sigma)\right).$$

Both distributions show the same mean and variance. Because both distributions are of even symmetry, all odd moments are zero. Thus the third moment (skewness) is also zero. However, the forth and all higher-order even moments of the two distributions are different.

15.3 Rotation and Scale Variant Texture Features

15.3.1 Local Orientation

As local orientation has already been discussed in detail in Chapter 13, we now only discuss some examples to illustrate the significance of local orientation for texture analysis. As this book contains only gray scale images, we only show coherence images of the local orientation.

Figure 15.4 shows the coherence measure for local orientation as defined in Section 13.3. This measure is one for an ideally oriented texture where the gray values change only in one direction, and zero for a distributed gray value structure. The coherency measure is close to one in the areas of the piece of shirt cloth with horizontal stripes (Fig. 15.4a) and in the dense parts of the dog fur (Fig. 15.4b). The orientation analysis of the curtain (Fig. 15.1a) results in an interesting coherency pattern (Fig. 15.4c). The coherency is high along the individual threads, but not at the corners where two threads cross each other, or in most of the areas in between. The coherency of the local orientation of the woodchip paper image (Fig. 15.1d) does not result in a uniform coherence image as this texture shows no predominant local orientation.

15.3.2 Local Wave Number

In Section 13.4 we discussed in detail the computation of the *local wave number* from a *quadrature filter pair* by means of either a *Hilbert filter* (Section 13.4.2) or quadrature filters (Section 13.4.5). In this section we apply these techniques to compute the characteristic scale of a texture using a *directiopyramidal decomposition* as a directional bandpass filter followed by Hilbert filtering.

The piece of shirt cloth in Fig. 15.5a shows distinct horizontal stripes in certain parts. This image is first bandpass filtered using the levels one and two of the vertical component of a directiopyramidal decomposition of the image (Fig. 15.5b). Figure 15.5c shows the estimate of the local wave number (component in vertical direction). All areas are masked out in which the amplitude of the corresponding structure (Fig. 15.5d) is not significantly higher than the noise level. In all areas with the horizontal stripes, a local wave number was computed. The histogram in Fig. 15.5e shows that the peak local wave number is about 0.133. This structure is sampled about 7.5 times per wavelength. Note the long tail of the distribution towards short wave numbers. Thus a secondary larger-scale structure is contained in the texture. This is indeed given by the small diagonal stripes.

Figure 15.6 shows the same analysis for a textured wood surface. This time the texture is more random. Nevertheless, it is possible to determine the local wave number. It is important, though, to mask out

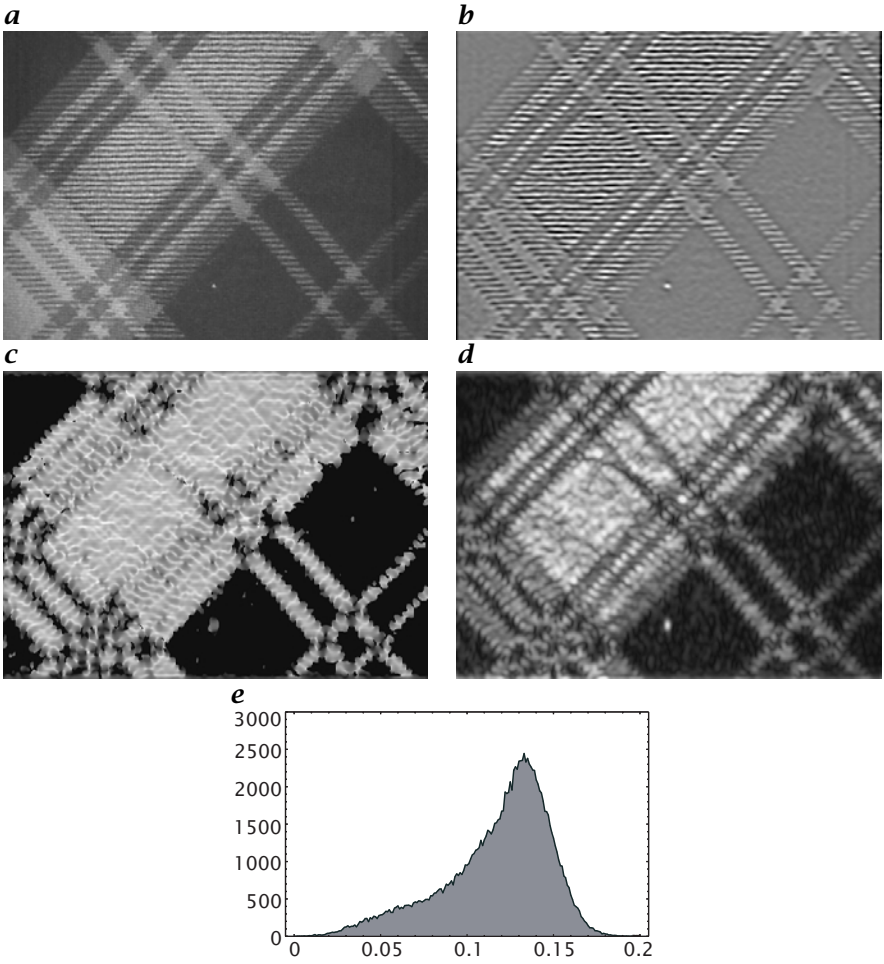

Figure 15.5: *Determination of the characteristic scale of a texture by computation of the local wave number:* ***a*** *original texture,* ***b*** *directional bandpass using the levels one and two of the vertical component of a directiopyramidal decomposition,* ***c*** *estimate of the local wave number (all structures below a certain threshold are masked to black),* ***d*** *amplitude of the local wave number, and* ***e*** *histogram of the local wave number distribution (units: number of periods per pixel).*

the areas in which no significant amplitudes of the bandpass filtered image are present. If the masking is not performed, the estimate of the local wave number will be significantly distorted. With the masking a quite narrow distribution of the local wave number is found with a peak at a wave number of 0.085.

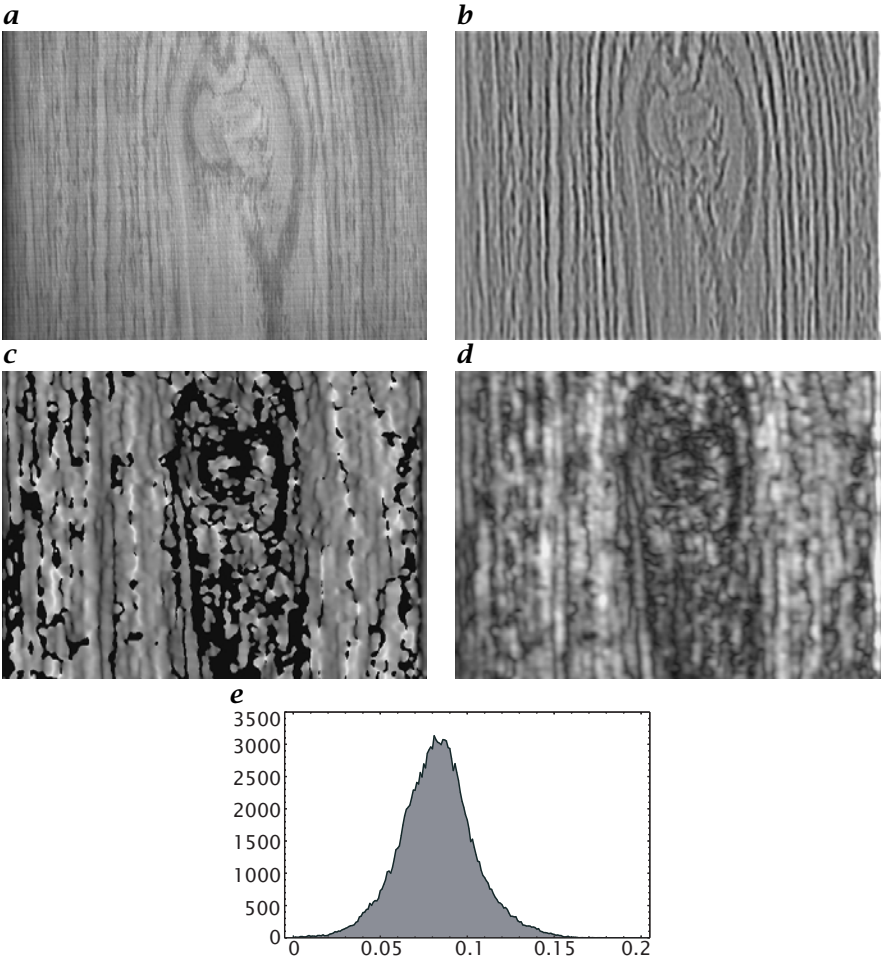

Figure 15.6: *Same as Fig. 15.5 applied to a textured wood surface.*

15.3.3 Pyramidal Texture Analysis

The Laplace pyramid is an alternative to the local wave number operator, because it results in a bandpass decomposition of the image. This decomposition does not compute a local wave number directly, but we can obtain a series of images which show the texture at different scales.

The variance operator takes a very simple form with a Laplace pyramid, as the mean gray value, except for the coarsest level, is zero:

$$\mathcal{V} = \mathcal{B}(\mathcal{L}^{(p)} \cdot \mathcal{L}^{(p)}). \tag{15.6}$$

Figure 15.7 demonstrates how the different textures from Fig. 15.1f appear at different levels of the Laplacian pyramid. In the two finest

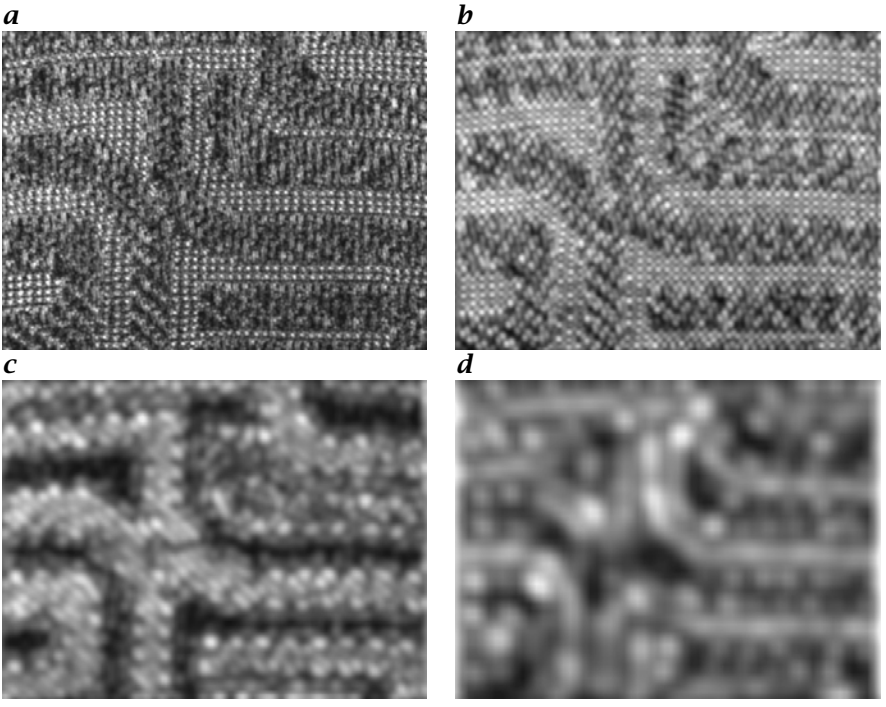

Figure 15.7: *Application of the variance operator to levels 0 to 3 of the Laplace pyramid of the image from Fig. 15.1f.*

scales at the zero and first level of the pyramid (Fig. 15.7a, b), the variance is dominated by the texture itself. The most pronounced feature is the variance around the dot-shaped stitches in one of the two textures.

At the second level of the Laplacian pyramid (Fig. 15.7c), the dot-shaped stitches are smoothed away and the variance becomes small in this texture while the variance is still significant in the regions with the larger vertically and diagonally oriented stitches. Finally, the third level (Fig. 15.7d) is too coarse for both textures and thus dominated by the edges between the two texture regions because they have a different mean gray value.

The Laplace pyramid is a very well adapted data structure for the analysis of hierarchically organized textures which may show different characteristics at different scales, as in the example of the curtain discussed in Section 15.1. In this way we can apply such operators as local variance and local orientation at each level of the pyramid. The simultaneous application of the variance and local orientation operators at multiple scales gives a rich set of features, which allows even complex hierarchically organized textures to be distinguished. It is important to

note that application of these operations on all levels of the pyramid only increases the number of computations by a factor of 4/3 for 2-D images.

15.4 Further Readings‡

The textbooks of Jain [86, Section 9.11] and Pratt [142, Chapter 17] deal also with texture analysis. Further references for texture analysis are the monography of Rao [146], the handbook by Jähne et al. [83, Vol. 2, Chapter 12], and the proceedings of the workshop on texture analysis, edited by Burkhardt [13].

Part IV

Image Analysis

16 Segmentation

16.1 Introduction

All image processing operations discussed in the preceding chapters aimed at a better recognition of objects of interest, i. e., at finding suitable local features that allow us to distinguish them from other objects and from the background. The next step is to check each individual pixel to see whether it belongs to an object of interest or not. This operation is called *segmentation* and produces a *binary image*. A pixel has the value one if it belongs to the object; otherwise it is zero. Segmentation is the operation at the threshold between *low-level image processing* and *image analysis*. After segmentation, we know which pixel belongs to which object. The image is parted into regions and we know the discontinuities as the boundaries between the regions. After segmentation, we can also analyze the shape of objects with operations such as those discussed in Chapter 19.

In this chapter, we discuss several types of elementary segmentation methods. Basically we can think of several basic concepts for segmentation. Pixel-based methods (Section 16.2) only use the gray values of the individual pixels. Region-based methods (Section 16.4) analyze the gray values in larger areas. Finally, edge-based methods (Section 16.3) detect edges and then try to follow them. The common limitation of all these approaches is that they are based only on local information. Even then they use this information only partly. Pixel-based techniques do not even consider the local neighborhood. Edge-based techniques look only for discontinuities, while region-based techniques analyze homogeneous regions. In situations where we know the geometric shape of an object, *model-based segmentation* can be applied (Section 16.5). We discuss an approach to the Hough transform that works directly from gray scale images (Section 16.5.3).

16.2 Pixel-Based Segmentation

Point-based or *pixel-based segmentation* is conceptually the simplest approach we can take for segmentation. We may argue that it is also the best approach. Why? The reason is that instead of trying a complex segmentation procedure, we should rather first use the whole palette

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2
Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

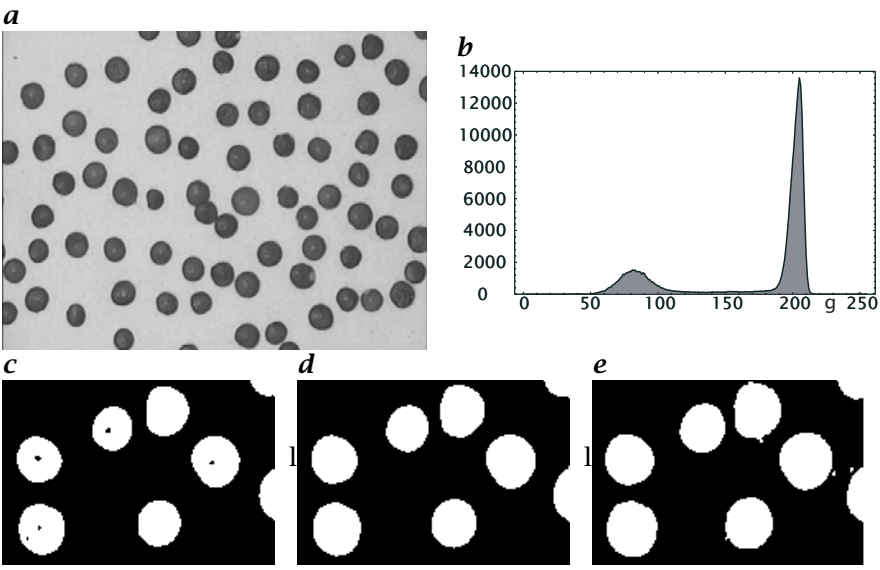

Figure 16.1: *Segmentation with a global threshold:* ***a*** *original image;* ***b*** *histogram;* ***c–e*** *upper right sector of* ***a*** *segmented with global thresholds of 110, 147, and 185, respectively.*

of techniques we have discussed so far in this book to extract those features that characterize an object in a unique way *before* we apply a segmentation procedure. It is always better to solve the problem at its root. If an image is unevenly illuminated, for instance, the first thing to do is to optimize the illumination of the scene. If this is not possible, the next step would be to identify the unevenness of the illumination system and to use corresponding image processing techniques to correct it. One possible technique has been discussed in Section 10.3.2.

If we have found a good feature to separate the object from the background, the histogram of this feature will show a *bimodal distribution* with two distinct maxima as in Fig. 16.1b. We cannot expect that the probability for gray values between the two peaks will be zero. Even if there is a sharp transition of gray values at the edge of the objects, there will always be some intermediate values due to a nonzero *point spread function* of the optical system and sensor (Sections 7.6.1 and 9.2.1). The smaller the objects are, the more area in the image is occupied by intermediate values filling the histograms in between the values for object and background (Fig. 16.1b).

How can we find an optimum threshold in this situation? In the case shown in Fig. 16.1, it appears to be easy because both the background and the object show rather uniform gray values. Thus we obtain a good segmentation for a large range of thresholds, between a low threshold of

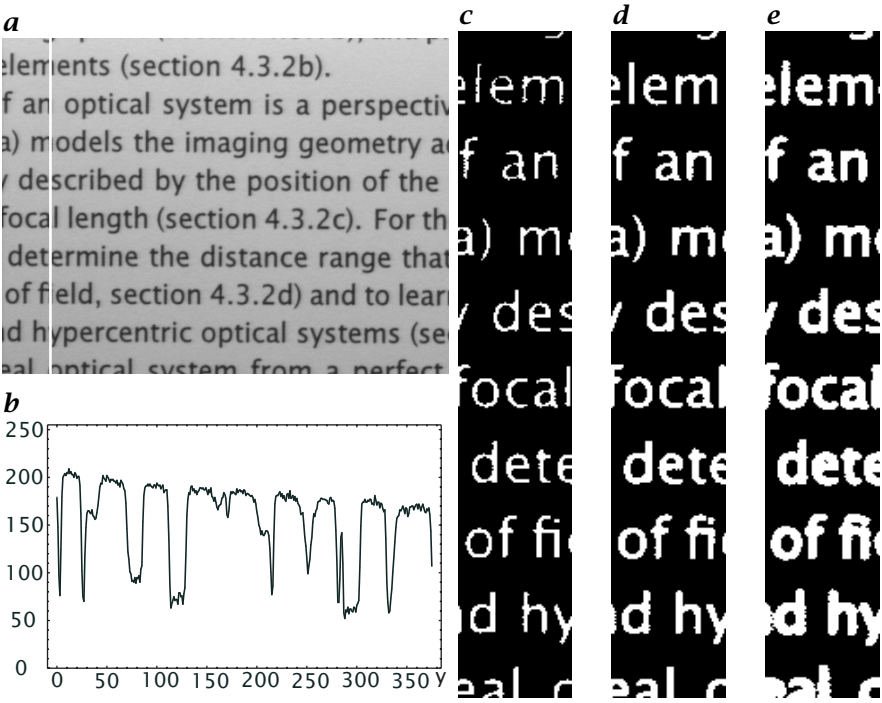

Figure 16.2: *Segmentation of an image with a graded background:* ***a*** *original image;* ***b*** *profile of column 55 (as marked in* ***a****);* ***c–e*** *first 64 columns of* ***a*** *segmented with global thresholds of 90, 120, and 150, respectively.*

110, where the objects start to get holes (Fig. 16.1c), and a high threshold of 185, close to the value of the background, where some background pixels are detected as object pixels.

However, a close examination of Fig. 16.1c–e reveals that the size of the segmented objects changes significantly with the level of the threshold. Thus it is critical for a bias-free determination of the geometrical features of an object to select the correct threshold. This cannot be performed without knowledge about the type of the edge between the object and the background. In the simple case of a symmetrical edge, the correct threshold is given by the mean gray value between the background and the object pixels.

This strategy fails as soon as the background is not uniform, or if objects with different gray values are contained in the image (Figs. 16.2 and 16.3). In Fig. 16.2b, the segmented letters are thinner in the upper, brighter part of the image. Such a bias is acceptable for some applications such as the recognition of typeset letters. However, it is a serious flaw for any gauging of object sizes and related parameters.

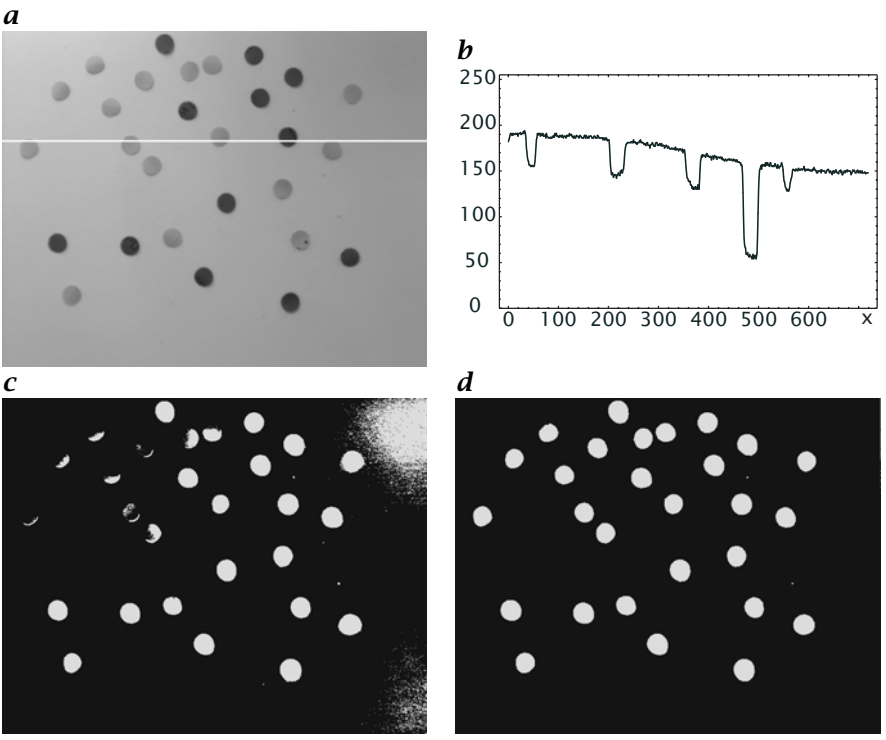

Figure 16.3: *Segmentation of an image with an uneven illumination:* ***a*** *original image with inhomogeneous background illumination (for histogram, see Fig. 10.10b);* ***b*** *profile of row 186 (as marked in* ***a****);* ***c*** *and* ***d*** *segmentation results with an optimal global threshold of the images in* ***a*** *before and after the image is first corrected for the inhomogeneous background (Fig. 10.10c), respectively.*

Figure 16.3a shows an image with two types of circles; both are circles but of different brightness. The radiance of the brighter circles comes close to the background. Indeed, a histogram (Fig. 10.10b) shows that the gray values of these brighter circles no longer form a distinct maximum but overlap with the wide distribution of the background.

Consequently, the global thresholding fails (Fig. 16.3c). Even with an optimal threshold, some of the background in the right upper and lower corners are segmented as objects and the brighter circles are still segmented only partly. If we first correct for the inhomogeneous illumination as illustrated in Fig. 10.10, all objects are segmented perfectly (Fig. 16.3d). We still have the problem, however, that the areas of the dark circles are too large because the segmentation threshold is too close to the background intensity.

16.3 Edge-Based Segmentation

16.3.1 Principle

We have seen in Section 16.2 that even with perfect illumination, pixel-based segmentation results in a bias of the size of segmented objects when the objects show variations in their gray values (Figs. 16.2 and 16.3). Darker objects will become too small, brighter objects too large. The size variations result from the fact that the gray values at the edge of an object change only gradually from the background to the object value. No bias in the size occurs if we take the mean of the object and the background gray values as the threshold. However, this approach is only possible if all objects show the same gray value or if we apply different thresholds for each objects.

An *edge-based segmentation* approach can be used to avoid a bias in the size of the segmented object without using a complex thresholding scheme. Edge-based segmentation is based on the fact that the position of an edge is given by an extreme of the first-order derivative or a zero crossing in the second-order derivative (Fig. 12.1). Thus all we have to do is to search for local maxima in the edge strength and to trace the maximum along the edge of the object.

16.3.2 Bias by Uneven Illumination

In this section, we study the bias of various segmentation techniques by a nonuniform background and varying object brightness. We assume that the object edge can be modeled adequately by a step edge that is blurred by a point spread function $h(x)$ with even symmetry. For the sake of simplicity, we model the 1-D case. Then the brightness of the object in the image with an edge at the origin can be written as

$$g(x) = g_0 \int_{-\infty}^{x} h(x)\mathrm{d}x \quad \text{with} \quad \int_{-\infty}^{\infty} h(x)\mathrm{d}x = 1. \tag{16.1}$$

We further assume that the background intensity can be modeled by a parabolic variation of the form

$$b(x) = b_0 + b_1 x + b_2 x^2. \tag{16.2}$$

Then the total intensity in the image is given by

$$g(x) = g_0 \int_{-\infty}^{x} h(x)\mathrm{d}x + b_0 + b_1 x + b_2 x^2. \tag{16.3}$$

The first and second derivatives are

$$\begin{aligned} g_x(x) &= g_0 h(x) + b_1 + 2b_2 x, \\ g_{xx}(x) &= g_0 h_x(x) + 2b_2. \end{aligned} \tag{16.4}$$

Around the maximum, we can approximate the point spread function $h(x)$ by a parabola: $h(x) \approx h_0 - h_2 x^2$. Then

$$\begin{aligned} g_x(x) &\approx g_0 h_0 - g_0 h_2 x^2 + b_1 + 2 b_2 x, \\ g_{xx}(x) &\approx -2 g_0 h_2 x + 2 b_2. \end{aligned} \tag{16.5}$$

The position of the edge is given as the zero of the second derivative. Therefore the bias in the edge position estimation, x_b, is given from Eq. (16.5) as

$$x_b \approx \frac{b_2}{g_0 h_2}. \tag{16.6}$$

From Eq. (16.6) we can conclude:

1. Edge-based segmentation shows no bias in the edge position even if the background intensity is sloped.
2. Edge-based segmentation shows no bias with the intensity g_0 of the edge as it is the case with intensity-based segmentation (Section 16.2).
3. Edge-based segmentation is only biased by a curvature in background intensity. The bias is directly related to the ratio of the curvature in the background intensity to the maximum curvature of the point spread function. This means that the bias is higher for blurred edges. The bias is also inversely proportional to the intensity of the object and thus seriously affects only objects with weak contrast.

16.3.3 Edge Tracking

Edge-based segmentation is a sequential method. In contrast to pixel-based and most region-based segmentations, it cannot be performed in parallel on all pixels. The next step to be performed rather depends on the results of the previous steps. A typical approach is as described in the following. The image is scanned line by line for maxima in the magnitude of the gradient. When a maximum is encountered, a *tracing algorithm* tries to follow the maximum of the gradient around the object until it reaches the starting point again. Then a search begins for the next maximum in the gradient. Like region-based segmentation, edge-based segmentation takes into account that an object is characterized by adjacent pixels.

16.4 Region-Based Segmentation

16.4.1 Principles

Region-based methods focus our attention on an important aspect of the segmentation process we missed with point-based techniques. There we classified a pixel as an object pixel judging solely on its gray value

independently of the context. This meant that isolated points or small areas could be classified as object pixels, disregarding the fact that an important characteristic of an object is its *connectivity*.

In this section we will not discuss such standard techniques as split-and-merge or region-growing techniques. Interested readers are referred to Rosenfeld and Kak [157] or Jain [86]. Here we discuss rather a technique that aims to solve one of the central problems of the segmentation process.

If we use not the original image but a feature image for the segmentation process, the features represent not a single pixel but a small neighborhood, depending on the mask sizes of the operators used. At the edges of the objects, however, where the mask includes pixels from both the object and the background, any feature that could be useful cannot be computed. The correct procedure would be to limit the mask size at the edge to points of either the object or the background. But how can this be achieved if we can only distinguish the object and the background after computation of the feature?

Obviously, this problem cannot be solved in one step, but only iteratively using a procedure in which feature computation and segmentation are performed alternately. In principle, we proceed as follows. In the first step, we compute the features disregarding any object boundaries. Then we perform a preliminary segmentation and compute the features again, now using the segmentation results to limit the masks of the neighborhood operations at the object edges to either the object or the background pixels, depending on the location of the center pixel. To improve the results, we can repeat feature computation and segmentation until the procedure converges into a stable result.

16.4.2 Pyramid Linking

Burt [15] suggested a *pyramid-linking* algorithm as an effective implementation of a combined segmentation feature computation algorithm. We will demonstrate it using the illustrative example of a noisy step edge (Fig. 16.4). In this case, the computed feature is simply the mean gray value. The algorithm includes the following steps:

1. *Computation of the Gaussian pyramid.* As shown in Fig. 16.4a, the gray values of four neighboring pixels are averaged to form a pixel on the next higher level of the pyramid. This corresponds to a smoothing operation with a box filter.
2. *Segmentation by pyramid-linking.* As each pixel contributes to either of two pixels on the higher level, we can now decide to which it most likely belongs. The decision is simply made by comparing the gray values and choosing the pixel next to it. The link is pictured in Fig. 16.4b by an edge connecting the two pixels. This procedure is

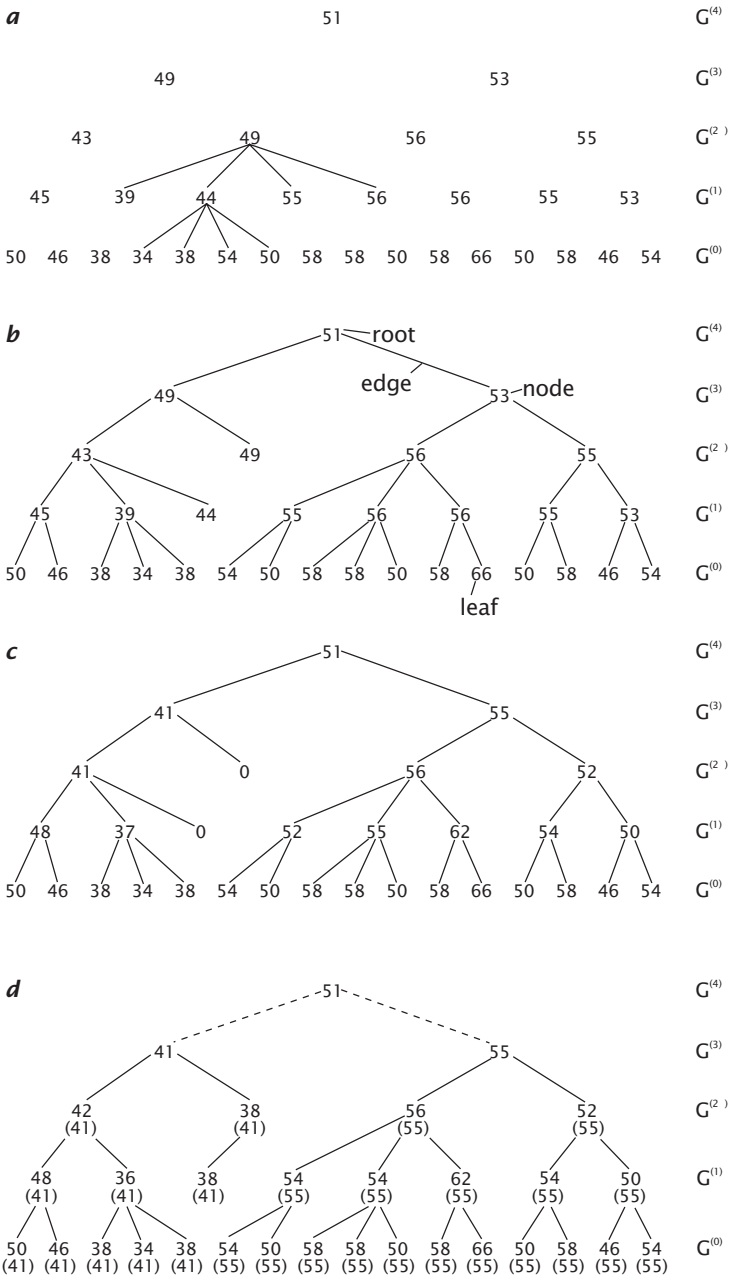

Figure 16.4: *Pyramid-linking segmentation procedure with a one-dimensional noisy edge:* ***a*** *computation of the Gaussian pyramid;* ***b*** *node-linking;* ***c*** *re-computation of the mean gray values;* ***d*** *final result after several iterations of steps* ***b*** *and* ***c*** *.*

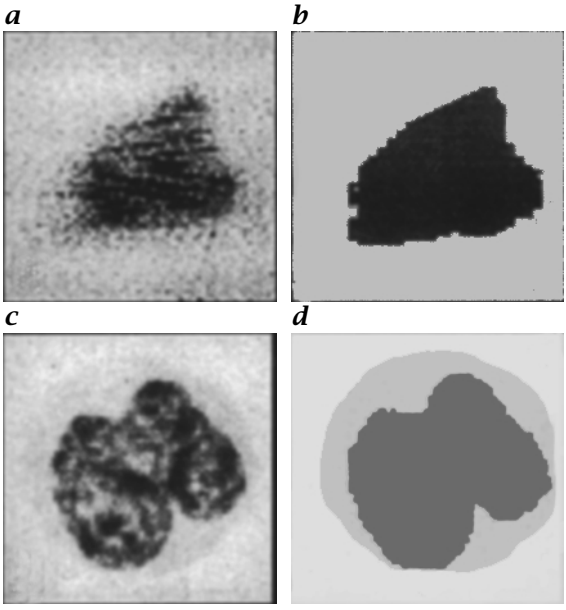

Figure 16.5: *Noisy* ***a*** *tank and* ***c*** *blood cell images segmented with the pyramid-linking algorithm in* ***b*** *two and* ***d*** *three regions, respectively; after Burt [15].*

repeated through all the levels of the pyramid. As a result, the links in the pyramid constitute a new data structure. Starting from the top of the pyramid, one pixel is connected with several pixels on the next lower level. Such a data structure is called a *tree* in computer science. The links are called *edges*; the data points are the gray values of the pixels and are denoted as *nodes* or *vertices*. The node at the highest level is called the *root* of the tree and the nodes with no further links are called the *leaves* of the tree. A node linked to a node at a lower level is denoted as the *father node* of this node. Correspondingly, each node linked to a node at a higher level is defined as the *son node* of this node.

3. *Averaging of linked pixels.* Next, the resulting link structure is used to recompute the mean gray values, now using only the linked pixels (Fig. 16.4c), i. e., the new gray value of each father node is computed as the average gray value of all the son nodes. This procedure starts at the lowest level and is continued through all the levels of the pyramid.

The last two steps are repeated iteratively until we reach a stable result shown in Fig. 16.4d. An analysis of the link-tree shows the result of the segmentation procedure. In Fig. 16.4d we recognize two *subtrees*, which have their roots in the third level of the pyramid. At the next lower level, four subtrees originate. But the differences in the gray values at

this level are significantly smaller. Thus we conclude that the gray value structure is obviously parted into two regions. Then we obtain the final result of the segmentation procedure by transferring the gray values at the roots of the two subtrees to the linked nodes at the lowest level. These values are shown as braced numbers in Fig. 16.4d.

The application of the pyramid-linking segmentation algorithm to two-dimensional images is shown in Fig. 16.5. Both examples illustrate that even very noisy images can be successfully segmented with this procedure. There is no restriction on the form of the segmented area.

The pyramid-linking procedure merges the segmentation and the computation of mean features for the objects extracted in an efficient way by building a tree on a pyramid. It is also advantageous that we do not need to know the number of segmentation levels beforehand. They are contained in the structure of the tree. Further details of pyramid-linking segmentation are discussed in Burt et al. [17] and Pietikäinen and Rosenfeld [138].

16.5 Model-Based Segmentation

16.5.1 Introduction

All segmentation techniques discussed so far utilize only local information. In Section 1.6 (Fig. 1.16) we noted the remarkable ability of the human vision system to recognize objects even if they are not completely represented. It is obvious that the information that can be gathered from local neighborhood operators is not sufficient to perform this task. Instead we require specific knowledge about the geometrical shape of the objects, which can then be compared with the local information.

This train of thought leads to *model-based segmentation*. It can be applied if we know the exact shape of the objects contained in the image. We consider here only the simplest case: straight lines.

16.5.2 Parameter Space; Hough Transform

The approach discussed here detects lines even if they are disrupted by noise or are only partially visible. We start by assuming that we have a segmented image that contains lines of this type. The fact that points lie on a straight line results in a powerful constraint that can be used to determine the parameters of the straight line. For all points $[x_n, y_n]^T$ on a straight line, the following condition must be met:

$$y_n = a_0 + a_1 x_n, \tag{16.7}$$

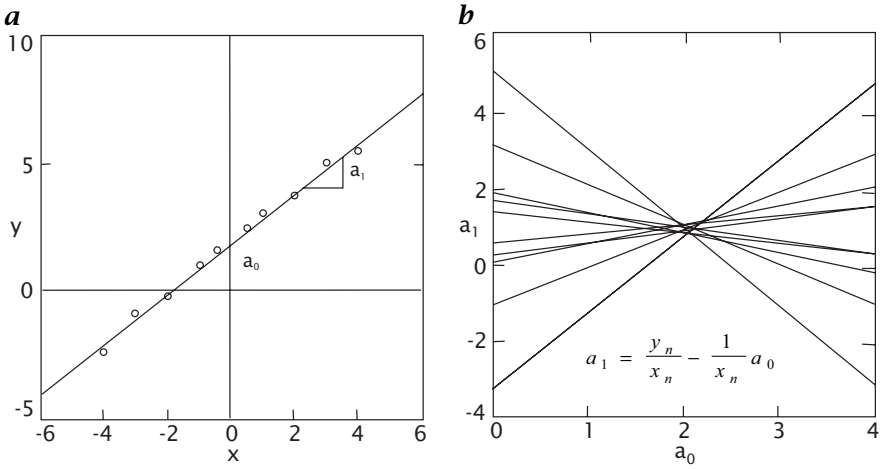

Figure 16.6: *Hough transform for straight lines: the $[x, y]^T$ data space (**a**) is mapped onto the $[a_0, a_1]^T$ model space (**b**).*

where a_0 and a_1 are the offset and slope of the line. We can read Eq. (16.7) also as a condition for the parameters a_0 and a_1:

$$a_1 = \frac{y_n}{x_n} - \frac{1}{x_n} a_0. \tag{16.8}$$

This is again the equation for a line in a new space spanned by the parameters a_0 and a_1. In this space, the line has the offset y_n/x_n and a slope of $-1/x_n$.

With one point given, we already cease to have a free choice of a_0 and a_1 as the parameters must satisfy Eq. (16.8).

The space spanned by the model parameters a_0 and a_1 is called the *model space*. Each point reduces the model space to a line. Thus, we can draw a line in the model space for each point in the data space, as illustrated in Fig. 16.6. If all points lie on a straight line in the data space, all lines in the model space meet in one point which gives the parameters a_0 and a_1 of the lines. As a line segment contains many points, we obtain a reliable estimate of the two parameters of the line. In this way, a line in the data space is mapped onto a point in the model space. This transformation from the data space to the model space via a model equation is called the *Hough transform*. It is a versatile instrument to detect lines even if they are disrupted or incomplete.

In practical applications, the well-known equation of a straight line given by Eq. (16.7) is not used. The reason is simply that the slope of a line may become infinite and is thus not suitable for a discrete model space. A better parameterization of a straight line is given by using two different parameters with finite values. One possibility is to take the

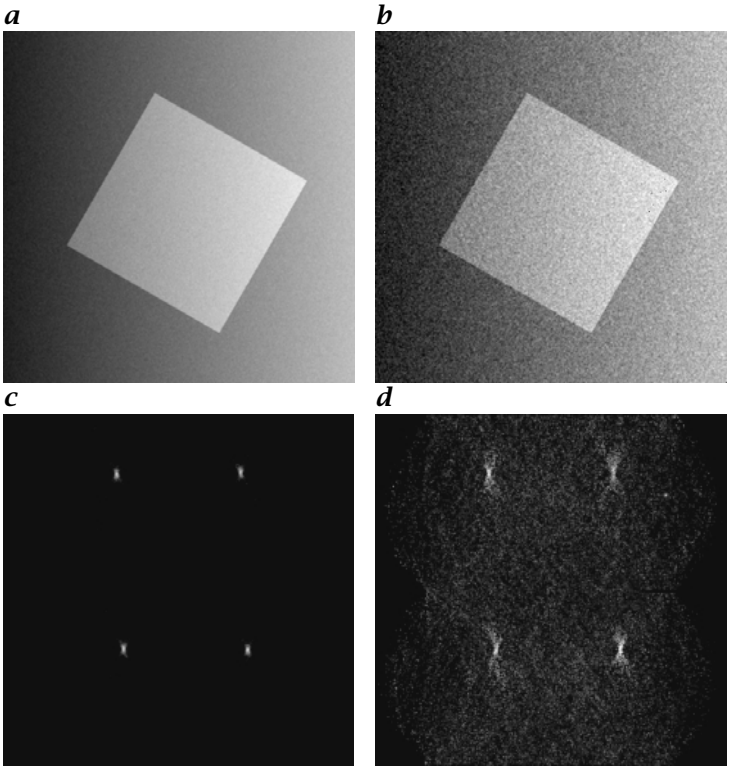

Figure 16.7: *Orientation-based fast Hough transform:* ***a*** *and* ***b*** *unevenly illuminated noisy squares;* ***c*** *and* ***d*** *Hough model space with the distance d (horizontal axis) and the angle θ (vertical axis) of the lines according to Eq. (16.9) for* ***a*** *and* ***b****, respectively.*

angle of the slope of the line and the distance of the line from the center of the coordinate system. With these two parameters, the equation of a straight line can be written as

$$\bar{\boldsymbol{n}}\boldsymbol{x} = d \quad \text{or} \quad x\cos\theta + y\sin\theta = d, \tag{16.9}$$

where $\bar{\boldsymbol{n}}$ is a vector normal to the line and θ the angle of this vector to the x axis of the image coordinate system.

The drawback of the Hough transform method for line detection is the high computational effort. For each point in the image, we must compute a line in the parameter space and increment each point in the model space through which the line passes.

16.5.3 Orientation-Based Fast Hough Transform

A significant speed-up of the Hough transform can be obtained by using additional information from low-level image processing. The analysis of local neighborhoods with the *structure tensor* method not only detects edges but also gives their slope. Therefore, we have two pieces of information for each point in the image if it lies on an edge: a point through which the edge passes and its orientation. This already completely describes the line.

Consequently, each point on a line in the image space corresponds no longer to a line — as discussed in Section 16.5.2 — but to a single point in the parameter space. The one-to-one correspondence considerably speeds up the computation of the Hough transform. For each point in the image, we only need to add *one* point to the parameter space.

An application of the orientation-based Hough transform is demonstrated in Fig. 16.7. Figure 16.7a, b shows noisy images with a square. To extract the edges of the rectangle, no segmentation is required. We just compute the components of the structure tensor with the techniques described in Section 13.3.6. Then for each point in the image, θ and d are computed according to Eq. (16.9).

As a weighting factor for the contribution of a point to the parameter space, we use the length of the orientation vector. In this way, points are weighted according to the certainty measure for the local orientation and thus the edge strength.

In the Hough parameter space (Fig. 16.7c and d), four clusters show up, corresponding to the four different lines of the square. The clusters occur in pairs as two lines are parallel to each other and differ only by the distance to the center of the image. Note how well the technique works even at high noise levels.

16.6 Further Readings‡

Pitas [139, Chapter 6] and Umbaugh [186, Section 2.4] deal with various standard algorithms for segmentation.

17 Regularization and Modeling

17.1 Unifying Local Analysis and Global Knowledge

The model-based segmentation technique discussed in Section 16.5 is a first step toward integrating global information into the process of object recognition. It is an inflexible technique, however, as it requires an exact parameterization of the objects to be detected. For real objects, it is often not possible to establish such an explicit type of model.

In this chapter, we discuss very general approaches to link local with global information that does not require an explicit model of the object. Instead it uses flexible constraints to include information of global type. The basic idea is to balance two counteracting requirements. On the one side, the model should reproduce the given image data as close as possible. This requirement is known as the *similarity constraint*. On the other side, the modeled data should meet some general global constraints that can be inferred from the general knowledge about the observed scene. In the simplest case this could be a *smoothness constraint*.

Generally, it is not possible to obtain an exact solution. Because all real-world image data incorporate a certain uncertainty, an exact fit of the data makes no sense. We rather expect a certain deviation of the computed model data from the image data that can be compared with the expected standard deviation of the noise contained in the data.

Thus we end up with a *global optimization* problem. Both kinds of constraint must be combined in an appropriate way to find a solution that has a minimum error with a given error norm.

This general approach can be applied to a wide range of image analysis problems including such diverse tasks as

- *restoration* of images degraded by the image formation process (Chapter 7),
- computation of *depth maps* from *stereo images* or any other imaging sensor based on triangulation techniques (Chapter 8.2),
- computation of *depth maps* from *shape from shading* or *photometric stereo* (Chapter 8.5),
- *reconstruction* of images from 3-D imaging techniques such as *tomography* (Section 8.6) that deliver no direct images,

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2

Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

- computation of motion or *displacement vector fields* from image sequences (Chapter 14),
- partition of images into regions (*segmentation*, Chapter 16), and
- computation of object boundaries (*active contours* or *snakes*).

Most of the features to be computed are scalar fields, but some of them, such as the motion field or surface normals, are vector fields. Therefore it is useful to extend the image modeling method to vector quantities.

Before we start, it is useful to consider the purpose and limits of modeling (Section 17.2). After detailing the general approach of variational image modeling in Section 17.3, we will discuss in Section 17.4 the important question discontinuities can be adequately incorporated into global smoothness constraints. The variational approach results in partial differential equations that are equivalent to transport equations including diffusion and reaction. Thus the discussion of diffusion models in Section 17.5 casts another interesting view on the problem of image modeling.

In the second part of this chapter, we turn to the discrete part of image modeling and see that it can be understood as a *discrete inverse problem* (Section 17.6). Electrical networks serve as an illustrative example (Section 17.7). In Section 17.8 we finally show with the example of *inverse filtering* how inverse problems can be solved efficiently.

17.2 Purpose and Limits of Models

The term *model* reflects the fact that any natural phenomenon can only be described to a certain degree of accuracy and correctness. It is one of the most powerful principles throughout all natural sciences to seek the simplest and most general description that still describes the observations with minimum deviations. A handful of basic laws of physics describe an enormously wide range of phenomena in a quantitative way.

Along the same lines, models are a useful and valid approach for image processing tasks. However, models must be used with caution. Even if the data seem to be in perfect agreement with the model assumptions, there is no guarantee that the model assumptions are correct.

Figure 17.1 shows an illustrative example. The model assumptions include a flat black object lying on a white background that is illuminated homogeneously (Fig. 17.1a). The object can be identified clearly by low gray values in the image, and the discontinuities between the high and low values mark the edges of the object.

If the black object has a non-negligible thickness, however, and the scene is illuminated by an oblique parallel light beam (Fig. 17.1c), we receive exactly the same type of profile as for Fig. 17.1a. Thus, we do

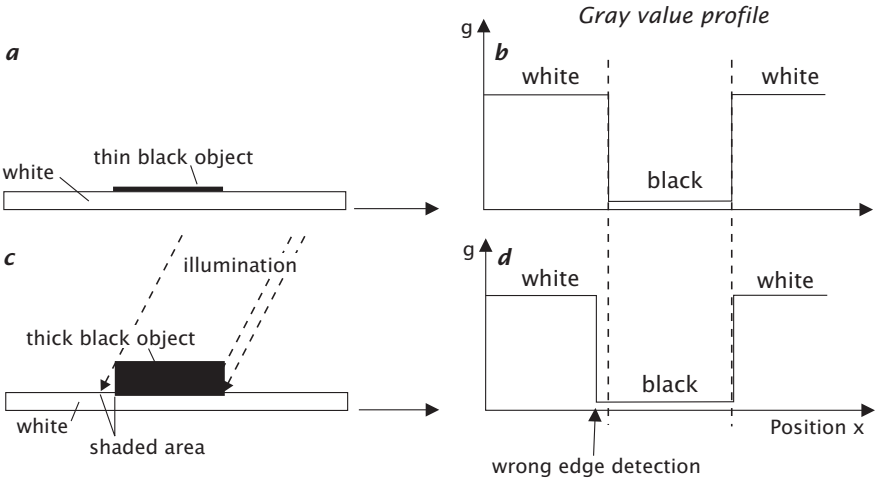

Figure 17.1: *Demonstration of a systematic error which cannot be inferred from the perceived image.* ***a*** *,* ***c*** *sketch of the object and illumination conditions;* ***b*** *and* ***d*** *resulting gray value profiles for* ***a*** *and* ***c*** *, respectively.*

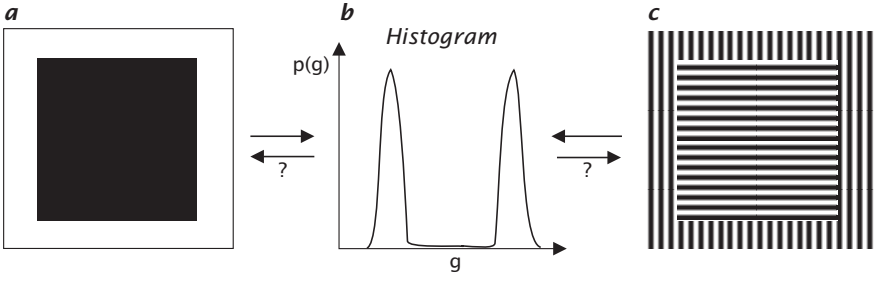

Figure 17.2: *Demonstration of a systematic deviation from a model assumption (object is black, background white) that cannot be inferred from the image histogram.*

not detect any deviation from the model assumption. Still only the right edge is detected correctly. The left edge is shifted to the left because of the shadowed region resulting in an image too large for the object.

Figure 17.2 shows another case. A black flat object fills half of the image on a white background. The histogram (the distribution of the gray values) clearly shows a bimodal shape with two peaks of equal height. This tells us that basically only two gray values occur in the image, the lower being identified as the black object and the higher as the white background, each filling half of the image.

This does not mean, however, that any bimodal histogram stems from an image where a black object fills half of the image against a white background. Many other interpretations are possible. For instance, also a white object could be encountered on a black background. The same bimodal histogram is also gained from an image in which both the object and the background are striped black and white. In the latter case, a segmentation procedure that allocates all pixels below a certain threshold to the object and the others to the background would not extract the desired object but the black stripes. This simple procedure only works if the model assumption is met that the objects and the background are of uniform brightness.

The two examples discussed above clearly demonstrate that even in simple cases we can run into situations where the model assumptions appear to be met — as judged by the image or quantities derived from the image such as histograms — but actually are not. While it is quite easy to see the failure of the model assumption in these simple cases, this may be more difficult if not impossible in more complex cases.

17.3 Variational Image Modeling†

As discussed in the introduction (Section 17.1), a mathematically well-founded approach to image modeling requires the setup of a model function and an error functional that measures the residual deviations of the measured data from the computed model data.

For image segmentation, a suitable modeling function could be a piecewise flat target function $f(\boldsymbol{x})$. Regions with constant values correspond to segmented objects and discontinuities to object boundaries. The free parameters of this model function would be gray values in the different regions and the boundaries between the regions. The boundaries between the objects and the gray values of the regions should be varied in such a way that the deviation between the model function $f(\boldsymbol{x})$ and the image data $g(\boldsymbol{x})$ are minimal.

The global constraints of this segmentation example are rather rigid. Smoothness constraints are more general. They tend to minimize the spatial variations of the feature. This concept is much more general than using a kind of fixed model saying that the feature should be constant as in the above segmentation example or vary only linearly.

Such global constraints can be handled in a general way using *variation calculus*. Before we turn to the application of variation calculus in image modeling, it is helpful to start with a simpler example from physics.

17.3.1 A Simple Example From Physics

Variation calculus has found widespread application throughout the natural sciences. It is especially well known in physics. All basic concepts of theoretical physics can be formulated as extremal principles. Probably the best known is *Hamilton's principle* which leads to the Lagrange equation in *theoretical mechanics* [53].

As a simple example, we discuss the motion of a mass point. Without external forces, the mass point will move with constant speed. The higher the mass, the more force is required to change its speed. Thus the mass tends to smoothen the velocity when the particle is moving through a spatially and temporally varying potential field $V(x,t)$ that applies the force $F = V_x(x,t)$ to the particle.

Hamilton's principle says that the motion follows a path for which the following integral is extreme:

$$\int_{t_1}^{t_2} \frac{1}{2} m\, x_t^2 - V(x,t)\,\mathrm{d}t. \tag{17.1}$$

The temporal derivative of x is denoted in Eq. (17.1) by x_t. The function in the integral is known as the *Lagrange function* $L(x, x_t, t)$. The Lagrange function depends on the position x and the time t via the potential $V(x,t)$ and on the temporal derivative of the position, i. e., the velocity, via the kinetic energy $m\, x_t^2/2$ of the mass point.

The above integral equation is solved by the *Euler-Lagrange equation*

$$\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial x_t} = 0 \quad \text{or short} \quad L_x - \frac{\mathrm{d}}{\mathrm{d}t} L_{x_t} = 0. \tag{17.2}$$

By this equation, the integral equation (Eq. (17.1)) can be converted into a differential equation for a given Lagrange function.

As an illustrative example we compute the motion of a mass point in a harmonic potential $V(x) = \epsilon x^2/2$. The Lagrange function of this system is

$$L(x, x_t, t) = T - V = \frac{1}{2} m (x_t)^2 - \frac{1}{2}\epsilon x^2. \tag{17.3}$$

The derivatives of the Lagrange function are

$$\frac{\partial L}{\partial x} = -\epsilon x, \quad \frac{\partial L}{\partial x_t} = m\, x_t, \quad \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial x_t} = m\, x_{tt}. \tag{17.4}$$

From the Euler equation (Eq. (17.2)) we obtain the simple second-order differential equation

$$m\, x_{tt} + \epsilon x = 0. \tag{17.5}$$

This second-order differential equation describes a harmonic oscillation of the mass point in the potential with a circular frequency $\omega = \sqrt{\epsilon/m}$.

17.3.2 Spatial and Spatiotemporal Variation Problems

In image processing it is required to formulate the variation problem for spatially and temporally varying variables. The path of the mass point $x(t)$, a scalar function, has to be replaced by a spatial function or spatiotemporal $f(\boldsymbol{x})$, i. e., by a scalar vector function of a vector variable. For image sequences, one of the components of $\boldsymbol{x}$ is the time t.

Consequently, the Lagrange function now depends on the vector variable $\boldsymbol{x}$. Furthermore, it will not only be a function of $f(\boldsymbol{x})$ and $\boldsymbol{x}$ explicitly. There will be additional terms depending on the spatial (and possibly temporal) partial derivatives of f. They are required as soon as we demand that f at a point should be dependent on f in the neighborhood. In conclusion, the general formulation of the *error functional* $\varepsilon(f)$ as a variation integral for g reads

$$\varepsilon(f) = \int_{\Omega} L\left(f, f_{x_p}, \boldsymbol{x}\right) \mathrm{d}x^W \rightarrow \text{minimum}. \tag{17.6}$$

The area integral is calculated over a certain image domain $\Omega \in \mathbb{R}^W$. Equation (17.6) already contains the knowledge that the extreme is a minimum. This results from the fact that f should show a minimum deviation from the given functions at certain points with additional constraints.

The corresponding Euler-Lagrange equation is:

$$L_f - \sum_{p=1}^{W} \partial_{x_p} L_{f_{x_p}} = 0. \tag{17.7}$$

The variational approach can also be extended to vectorial features such as the velocity in image sequences. Then, the Lagrange function depends on the vectorial feature $\boldsymbol{f} = [f_1, f_2, \ldots, f_W]^T$, the partial derivatives of each component f_i of the feature in all directions $(f_i)_{x_p}$, and explicitly on the coordinate $\boldsymbol{x}$:

$$\varepsilon(\boldsymbol{f}) = \int_{\Omega} L\left(\boldsymbol{f}, (f_i)_{x_p}, \boldsymbol{x}\right) \mathrm{d}x^W \rightarrow \text{minimum}. \tag{17.8}$$

From this equation, we obtain an Euler–Lagrange equation for each component f_i of the vectorial feature:

$$L_{f_i} - \sum_{p=1}^{W} \partial_{x_p} L_{(f_i)_{x_p}} = 0. \tag{17.9}$$

17.3.3 Similarity Constraints

The similarity term is used to make the modeled feature similar to the measured feature. For a simple segmentation problem, in which the objects can be distinguished by their gray value, the measured feature is the gray value itself and the similarity term S is given by

$$S(f, \boldsymbol{x}) = \|f(\boldsymbol{x}) - g(\boldsymbol{x})\|_n. \tag{17.10}$$

This simply means that the deviation between the modeled feature and the image measured with the L_n norm should be minimal. The most commonly used norm is the L_2 norm, leading to the well known *least squares* (LS) approach.

For a linear *restoration* problem, the original image $f(\boldsymbol{x})$ is degraded by a convolution operation with the point spread function of the degradation $h(\boldsymbol{x})$ (for further details, see Section 17.8). Thus the measured image $g(\boldsymbol{x})$ is given by

$$g(\boldsymbol{x}) = h(\boldsymbol{x}) * f(\boldsymbol{x}). \tag{17.11}$$

In order to obtain a minimum deviation between the measured and reconstructed images, the similarity term is

$$S(f, \boldsymbol{x}) = \|h(\boldsymbol{x}) * f(\boldsymbol{x}) - g(\boldsymbol{x})\|_n. \tag{17.12}$$

As a last example, we discuss the similarity constraint for motion determination. In Section 14.3.2 we discussed that the optical flow should meet the brightness constraint equation (14.9):

$$\boldsymbol{f}(\boldsymbol{x}, t)\nabla g(\boldsymbol{x}, t) + g_t(\boldsymbol{x}, t) = 0 \tag{17.13}$$

and used an approach that minimized the deviation from the optical flow in a least squares sense (Eq. (14.15)). With the L_n norm, we obtain the following similarity term:

$$S(\boldsymbol{f}, \boldsymbol{x}, t) = \|\boldsymbol{f}\,\nabla g + g_t\|_n. \tag{17.14}$$

This equation simply expresses that the continuity equation for the optical flow (Eq. (14.9)) should be satisfied as well as possible in a least squares sense. Note that the similarity now also depends explicitly on time, because the minimization problem is extended from images to space-time images.

From the following example, we will learn that similarity constraints alone are not of much use with the variational approach. We use the motion determination problem with the L_2-norm (least squares). With Eq. (17.14), the Lagrange function depends only on the optical flow $\boldsymbol{f}$. To compute the Euler-Lagrange equations, we only need to consider the

partial derivatives of the similarity term Eq. (17.14) with respect to the components of the optical flow, $\partial L/\partial f_i$:

$$L_{f_i} = 2\left(\boldsymbol{f}\,\nabla g + g_t\right) g_{x_i}. \tag{17.15}$$

Inserting Eq. (17.15) into the Euler-Lagrange equation (Eq. (17.9)) yields

$$\left(\boldsymbol{f}\,\nabla g + g_t\right) g_x = 0, \quad \left(\boldsymbol{f}\,\nabla g + g_t\right) g_y = 0, \tag{17.16}$$

or, written as a vector equation,

$$\left(\boldsymbol{f}\,\nabla g + g_t\right) \nabla g = \mathbf{0}. \tag{17.17}$$

These equations tell us that the optical flow cannot be determined when the spatial gradient of ∇g is a zero vector. Otherwise, they yield no more constraints than the continuity of the optical flow. This example nicely demonstrates the limitation of local similarity constraints. They only yield *isolated local* solutions without any constraints for the spatial variation of the optical flow. This results from the fact that the formulation of the problem does not include any terms connecting neighboring points. Thus, real progress requires inclusion of global constraints.

17.3.4 Global Smoothness Constraints

One of the most elementary global regularizers is smoothness. For many problems in image processing it makes sense to demand that a quantity to be modeled changes only slowly in space and time. For a segmentation problem this demand means that an object is defined just by the fact that it is a connected region with constant or only slowly changing features. Likewise, the depth of a surface and the velocity field of a moving object are continuous at least at most points.

Therefore, we now seek a suitable regularizer R to add to the Lagrange function to force spatially smooth solutions. Such a term requires spatial partial derivatives of the modeled feature. The simplest term, containing only first-order derivatives, for a scalar feature f in a 2-D image is

$$R\left(f_x, f_y\right) = \alpha^2\left(f_x^2 + f_y^2\right) = \alpha^2 |\nabla f|^2. \tag{17.18}$$

For a vector feature $\boldsymbol{f} = [f_1, f_2]^T$

$$R\left(\boldsymbol{f}_x, \boldsymbol{f}_y\right) = \alpha^2\left(|\nabla f_1|^2 + |\nabla f_2|^2\right). \tag{17.19}$$

In this additional term the partial derivatives emerge as a sum of squares. This means that we evaluate the *smoothness* term with the same

norm (L_2-norm, sum of least squares) as the similarity term. Moreover, in this formulation all partial derivatives are weighted equally. The factor α^2 indicates the relative weight of the smoothness term compared to the similarity term.

The complete least-squares error functional for motion determination including the similarity and smoothing terms is then given by

$$L\left(\boldsymbol{f}, \boldsymbol{f}_x, \boldsymbol{f}_y\right) = \left(\boldsymbol{f}\,\nabla g + g_t\right)^2 + \alpha^2\left(|\nabla f_1|^2 + |\nabla f_2|^2\right). \tag{17.20}$$

Inserting this Lagrange function into the Euler-Lagrange equation (17.9) yields the following differential equation:

$$\begin{aligned} \left(\nabla g\;\boldsymbol{f} + g_t\right) g_x \quad &-\alpha^2\left((f_1)_{xx} + (f_1)_{yy}\right) = \quad 0 \\ \left(\nabla g\;\boldsymbol{f} + g_t\right) g_y \quad &-\alpha^2\left((f_2)_{xx} + (f_2)_{yy}\right) = \quad 0, \end{aligned} \tag{17.21}$$

or summarized in a vector equation:

$$\underbrace{\left(\nabla g\;\boldsymbol{f} + \frac{\partial g}{\partial t}\right)}_{\text{similarity term}} \nabla g - \underbrace{\alpha^2 \Delta \boldsymbol{f}}_{\text{smoothness term}} = \boldsymbol{0}. \tag{17.22}$$

It is easy to grasp how the optical flow results from this formula. First, imagine that the intensity is changing strongly in a certain direction. The similarity term then becomes dominant over the smoothness term and the velocity will be calculated according to the local optical flow. In contrast, if the intensity change is small, the smoothness term becomes dominant. The local velocity will be calculated in such a manner that it is as close as possible to the velocity in the neighborhood. In other words, the flow vectors are interpolated from surrounding flow vectors.

This process may be illustrated further by an extreme example. Let us consider an object with a constant intensity moving against a black background. Then the similarity term vanishes completely inside the object, while at the border the velocity perpendicular to the border can be calculated just from this term.

This is an old and well-known problem in physics: the problem of how to calculate the potential function (without sinks and sources, $\Delta \boldsymbol{f} = \boldsymbol{0}$) with given boundary conditions at the edge of the object. This equation is known as the *Laplacian equation*. We can immediately conclude the form of the solution in areas where the similarity term is zero. As the second-order derivatives are zero, the first-order spatial derivatives are constant. This leads to a modeled feature f that changes linearly in space.

17.3.5 Elasticity Models‡

At this point of our discussion, it is useful to discuss an analogous physical problem that gives further insight how similarity and smoothing constraints balance each other. With a physical model these two terms correspond to two types of forces.

Again, we will use the example of optical flow determination. We regard the image as painted onto an *elastic membrane*. Motion will shift the membrane from image to image. Especially nonuniform motion causes a slight expansion or contraction of the membrane. The similarity term acts as an external force that tries to pull the membrane towards the corresponding *displacement vector* (DV). The inner elastic forces distribute these deformations continuously over the whole membrane, producing a smooth *displacement vector field* (DVF).

Let us first consider the external forces in more detail. It does not make much sense to set the deformations at those points where we can determine the DV to the estimated displacement without any flexibility. Instead we allow deviations from the expected displacements which may be larger, the more uncertain the determination of the DV is. Physically, this is similar to a pair of springs whose spring constant is proportional to the certainty with which the displacement can be calculated. The zero point of the spring system is set to the computed displacement vector. As the membrane is two-dimensional, two pairs of springs are required. The direction of the spring system is aligned according to the *local orientation* (Section 13.3). At an edge, only the displacement normal to the edge can be computed (*aperture problem*, Section 14.2.2). In this case, only one spring pair is required; a displacement parallel to the edge does not result in a restoring force.

The external spring forces are balanced by the inner elastic forces trying to even out the different deformations. Let us look again at the Euler-Lagrange equation of the optical flow (Eq. (17.22)) from this point of view. We can now understand this equation in the following way:

$$\underbrace{(\nabla g\, \boldsymbol{f} + g_t)\, \nabla g}_{\text{external force}} - \underbrace{\alpha^2 \Delta \boldsymbol{f}}_{\text{internal force}} = \boldsymbol{0}, \tag{17.23}$$

where α^2 is an *elasticity constant*. In the expression for the internal forces only second derivatives appear, because a constant gradient of the optical flow does not result in net inner forces.

The elasticity features of the membrane are expressed in a single constant. Further insight into the inner structure of the membrane is given by the Lagrange function (Eq. (17.18)):

$$L\left(\boldsymbol{f}, \boldsymbol{f}_{x_p}, \boldsymbol{x}\right) = \underbrace{\alpha^2 \left(|\nabla \boldsymbol{f}_1|^2 + |\nabla \boldsymbol{f}_2|^2\right)}_{T,\ \text{deformation energy}} + \underbrace{(\nabla g\, \boldsymbol{f} + g_t)^2}_{-V,\ \text{potential}}. \tag{17.24}$$

The Lagrange function is composed of the *potential* of the external force as it results from the continuity of the optical flow and an energy term related to the inner forces. This term is thus called *deformation energy*. This energy appears in place of the kinetic energy in the classical example of the Lagrange function for a mass point, as the minimum is not sought in time but in space.

The deformation energy may be split up into several terms which are closely related to the different modes of deformation:

$$T\left(\boldsymbol{f}_{x_p}\right) = \frac{1}{2}\left[\underbrace{\left((f_1)_x + (f_2)_y\right)^2}_{\text{dilation}} + \underbrace{\left((f_1)_x - (f_2)_y\right)^2 + \left((f_1)_y + (f_2)_x\right)^2}_{\text{shear}} + \underbrace{\left((f_1)_y - (f_2)_x\right)^2}_{\text{rotation}}\right]. \tag{17.25}$$

Clearly, the elasticity features of the membrane match the kinematics of motion optimally. Each possible deformation that may occur because of the different modes of 2-D motion on the image plane is equally weighted.

Physically, such a membrane makes no sense. The differential equation for a real physical membrane is different [39]:

$$\boldsymbol{f} - (\lambda + \mu)\boldsymbol{\nabla}(\boldsymbol{\nabla}\boldsymbol{u}) - \mu\Delta\boldsymbol{u} = \boldsymbol{0}. \tag{17.26}$$

The elasticity of a physical membrane is described by the two constants λ and μ. $\lambda = -\mu$ is not possible; as a result, the additional term $\nabla(\nabla\boldsymbol{u})$ (in comparison to the model membrane for the DVF) never vanishes. If there is no cross contraction, λ can only be zero.

With the membrane model, only the elongation is continuous, but not the first-order derivative. Discontinuities occur exactly at the points where external forces are applied to the membrane. This results directly from Eq. (17.22). A locally applied external force corresponds to a δ distribution in the similarity term. Integrating Eq. (17.22), we obtain a discontinuity in the first-order derivatives.

These considerations call into question the smoothness constraints considered so far. We know that the motion of planar surface elements does not result in such discontinuities. Smoothness of the first-order derivatives can be forced if we include second-order derivatives in the smoothness term (Eq. (17.22)) or the deformation energy (Eq. (17.24)). Physically, such a model is similar to a thin *elastic plate* that cannot be folded like a membrane.

17.4 Controlling Smoothness

Having discussed the basic properties of smoothness constraints we now turn to the question of how we can adequately treat spatial and temporal discontinuities with this approach. In a segmentation problem, the modeled feature will be discontinuous at the edge of the object. The same is true for the optical flow. The smoothness constraint as we have formulated it so far does not allow for discontinuities. We applied a global smoothness constraint and thus obtained a globally smooth field. Thus we need to develop methods that allow us to detect and to model discontinuities adequately.

We will first discuss the principal possibilities for varying the minimal problem within the chosen frame. To do so, we rewrite the integral equation for the minimal problem (Eq. (17.6)) using the knowledge about the meaning of the Lagrange function obtained in the last section:

$$\int_{\Omega} \underbrace{\left(S(\boldsymbol{f})\right.}_{\text{similarity term}} + \underbrace{\left.R(\boldsymbol{f}_{x_p})\right)}_{\text{smoothness term}} \mathrm{d}^W x \rightarrow \text{Minimum}. \tag{17.27}$$

In order to incorporate discontinuities, two approaches are possible:

1. *Limitation of integration area.* The integration area is one of the ways that the problem of discontinuities in the feature $\boldsymbol{f}$ may be solved. If the integration area includes discontinuities, incorrect values are obtained. Thus algorithms must be found that look for edges in $\boldsymbol{f}$ and, as a consequence, restrict the integration area to the segmented areas. Obviously, this is a difficult iterative procedure. First, the edges in the image itself do not necessarily coincide with the edges in the feature $\boldsymbol{f}$. Second, before calculating the feature field $\boldsymbol{f}$ only sparse information is available so that a partition is not possible.
2. *Modification of smoothness term.* Modification of the smoothness term is another way to solve the discontinuity problem. At points where a discontinuity is suspected, the smoothness constraint may be weakened or may even vanish. This allows discontinuities. Again this is an iterative algorithm. The smoothness term must include a control function that switches off the smoothness constraint in appropriate circumstances. This property is called *controlled smoothness* [181].

In the following, we discuss two approaches that modify the integration area for motion determination. The modification of the smoothing term is discussed in detail in Section 17.5.

Integration along Closed Zero-Crossing Curves. Hildreth [68] used the Laplace filtered image, and limited any further computations to zero crossings. This approach is motivated by the fact that zero crossings mark the gray value edges (Section 12.2), i. e., the features at which we can compute the velocity component normal to the edge. The big advantage of the approach is that the preselection of promising features considerably decreases any computation required.

By selecting the zero crossings, the smoothness constraint is limited to a certain contour line. This seems useful, as a zero crossing most likely belongs to an object but does not cross object boundaries. However, this is not necessarily the case. If a zero crossing is contained within an object, the velocity along the contour should show no discontinuities. Selecting a line instead of an area for the smoothness constraint

Figure 17.3: *Two images of a Hamburg taxi. The video images are from the Computer Science Department at Hamburg University and since then have been used as a test sequence for image sequence processing.*

changes the integration region from an area to the line integral along the contour s:

$$\oint \left\{ (\tilde{\boldsymbol{n}}\boldsymbol{f} - f_\perp)^2 + \alpha^2 \left[((f_1)_s)^2 + ((f_2)_s)^2 \right] \right\} \mathrm{d}\boldsymbol{s} \rightarrow \text{minimum}, \qquad (17.28)$$

where $\tilde{\boldsymbol{n}}$ is a unit vector normal to the edge and $f_\perp$ the velocity normal to the edge.

The derivatives of the velocities are computed in the direction of the edge. The component normal to the edge is given directly by the similarity term, while the velocity term parallel to the edge must be inferred from the smoothness constraint all along the edge. Hildreth [68] computed the solution of the linear equation system resulting from Eq. (17.28) iteratively using the method of conjugate gradients.

Despite its elegance, the edge-oriented method shows significant disadvantages. It is not certain that a zero crossing is contained within an object. Thus we cannot assume that the optical flow field is continuous along the zero crossing. As only edges are used to compute the optical flow field, only one component of the displacement vector can be computed locally. In this way, all features such as either gray value maxima or gray value corners which allow an unambiguous local determination of a displacement vector are disregarded.

Limitation of Integration to Segmented Regions. A region-oriented approach does not omit such points, but still tries to limit the smoothness within objects. Again, *zero crossings* could be used to separate the image into regions or any other basic segmentation technique (Chapter 16). Region-limited smoothness just drops the continuity constraint at the boundaries of the region. The simplest approach to this form of

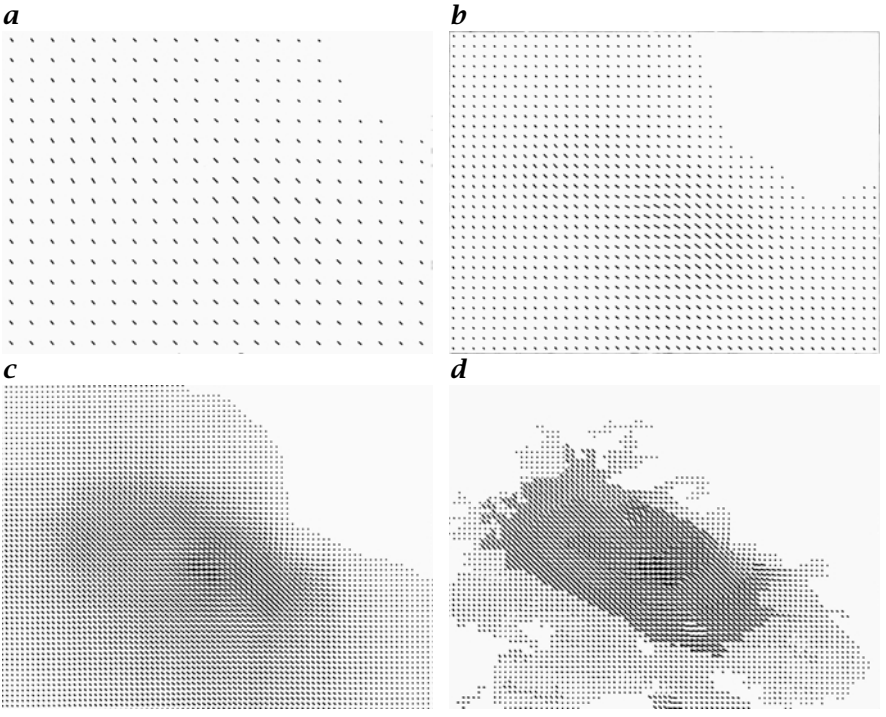

Figure 17.4: *Determination of the DVF in the taxi scene (Fig. 17.3) using the method of the dynamic pyramid:* ***a–c*** *three levels of the optical flow field using a global smoothness constraint;* ***d*** *final result of the optical flow using a region-oriented smoothness constraint; kindly provided by* M. Schmidt *and* J. Dengler, *German Cancer Research Center, Heidelberg.*

constraint is to limit the integration areas to the different regions and to evaluate them separately.

As expected, a region-limited smoothness constraint results in an optical flow field with discontinuities at the region's boundaries (Fig. 17.4d) which is in clear contrast to the globally smooth optical flow field in Fig. 17.4c. We immediately recognize the taxi by the boundaries of the optical flow.

We also observe, however, that the car is segmented further into regions with different optical flow field, as shown by the taxi plate on the roof of the car and the back and side windows. The small regions especially show an optical flow field significantly different from that in larger regions. Thus a simple region-limited smoothness constraint does not reflect the fact that there might be separated regions within objects. The optical flow field may well be smooth across these boundaries.

17.5 Diffusion Models

In this section, we take a new viewpoint of variational image modeling. The least-squares error functional for motion determination (17.20)

$$\left(\nabla g\, \boldsymbol{f} + \frac{\partial g}{\partial t}\right) \nabla g - \alpha^2 \Delta \boldsymbol{f} = \boldsymbol{0} \tag{17.29}$$

can be regarded as the stationary solution of a *diffusion-reaction system* with homogeneous diffusion if the constant α^2 is identified with a diffusion coefficient D:

$$\boldsymbol{f}_t = D\Delta \boldsymbol{f} - \left(\nabla g\, \boldsymbol{f} + \frac{\partial g}{\partial t}\right) \nabla g. \tag{17.30}$$

The standard instationary (partial) differential equation for homogeneous diffusion (see Eq. (5.10) in Section 5.2.1) is appended by an additional source term related to the similarity constraint. The source strength is proportional to the deviation from the optical flow constraint. Thus this term tends to shift the values for $\boldsymbol{f}$ to meet the optical flow constraint.

After this introductionary example, we can formulate the relation between a variational error functional and diffusion-reaction systems in a general way. The *Euler-Lagrange equation*

$$\sum_{p=1}^{W} \partial_{x_p} L_{f_{x_p}} - L_f = 0 \tag{17.31}$$

that minimizes the error functional for the scalar spatiotemporal function $f(\boldsymbol{x}), \boldsymbol{x} \in \Omega$

$$\varepsilon(f) = \int_{\Omega} L\left(f, f_{x_p}, \boldsymbol{x}\right) \mathrm{d}x^W \tag{17.32}$$

can be regarded as the steady state of the diffusion-reaction system

$$f_t = \sum_{p=1}^{W} \partial_{x_p} L_{f_{x_p}} - L_f. \tag{17.33}$$

In the following we shall discuss an aspect of modeling we have not touched so far — the local modification of the smoothness term. In the language of a diffusion model this means a locally varying diffusion coefficient in the first-hand term on the right side of Eq. (17.33). From the above discussion we know that to each approach of locally varying diffusion coefficient, a corresponding variational error functional exists that is minimized by the diffusion-reaction system.

In Section 5.2.1 we discussed a homogeneous diffusion process that generated a multiresolution representation of an image, known as the

linear *scale space*. If the smoothness constraint is made dependent on local properties of the image content such as the gradient, then the inhomogeneous diffusion process leads to the generation of a nonlinear scale space. With respect to modeling, the interesting point here is that a segmentation can be achieved without a similarity term.

17.5.1 Inhomogeneous Diffusion

The simplest approach to a spatially varying smoothing term that takes into account discontinuities is to reduce the diffusion coefficient at the edges. Thus, the diffusion coefficient is made dependent on the strength of the edges as given by the magnitude of the gradient

$$D(f) = D(|\nabla f|^2). \tag{17.34}$$

With a locally varying diffusion constant the diffusion-reaction system becomes

$$f_t = \nabla \left(D(|\nabla f|^2) \nabla f \right) - L_f. \tag{17.35}$$

Note that it is incorrect to write $D(|\nabla f|^2)\,\Delta f$. This can be seen from the derivation of the instationary diffusion equation in Section 5.2.1.

With Eq. (17.35) the regularization term R in the Lagrange function is

$$R = R(|\nabla f|^2), \tag{17.36}$$

where the diffusion coefficient is the derivative of the function R: $D = R'$. This can easily be verified by inserting Eq. (17.36) into Eq. (17.31).

Perona and Malik [136] used the following dependency of the diffusion coefficient on the magnitude of the gradient:

$$D = D_0 \frac{\lambda^2}{|\bigtriangledown f|^2 + \lambda^2}, \tag{17.37}$$

where λ is an adjustable parameter. For low gradients $|\bigtriangledown g| \ll \lambda$, D approaches D_0; for high gradients $|\bigtriangledown g| \gg \lambda$, D tends to zero.

As simple and straightforward as this idea appears, it is not without problems. Depending on the functionality of D on ∇g, the diffusion process may become unstable, even resulting in steepening of the edges. A safe way to avoid this problem is to use a regularized gradient obtained from a smoothed version of the image as shown by Weickert [195]. He used

$$D = 1 - \exp\left(-\frac{c_m}{(|\nabla (B^r * f)(\boldsymbol{x})|/\lambda)^m} \right). \tag{17.38}$$

This equation implies that for small magnitudes of the gradient the diffusion coefficient is constant. At a certain threshold of the magnitude of the gradient, the diffusion coefficient quickly decreases towards zero.

The higher the exponent m is, the steeper the transition. With the values used by Weickert [195], $m = 4$ and $c_4 = 3.31488$, the diffusion coefficient falls from 1 at $|\nabla g|/\lambda = 1$ to about 0.15 at $|\nabla g|/\lambda = 2$. Note that a regularized gradient has been chosen in Eq. (17.38), because the gradient is not computed from the image $g(\boldsymbol{x})$ directly, but from the image smoothed with a binomial smoothing mask $\mathcal{B}^p$. A properly chosen regularized gradient stabilizes the inhomogeneous smoothing process and avoids instabilities and steepening of the edges.

A simple explicit discretization of inhomogeneous diffusion uses regularized derivative operators as discussed in Section 12.5. In the first step, a gradient image is computed with the vector operator

$$\begin{bmatrix} \mathcal{D}_1 \\ \mathcal{D}_2 \end{bmatrix}. \tag{17.39}$$

In the second step, the gradient image is multiplied by the control image $\boldsymbol{R}$ that computes the diffusion coefficient according to Eq. (17.38) with $D_0 = 1$:

$$\begin{bmatrix} \mathcal{R} \cdot \mathcal{D}_1 \\ \mathcal{R} \cdot \mathcal{D}_2 \end{bmatrix}. \tag{17.40}$$

The control image $\mathcal{R}$ is one in constant regions and drops towards small values at the edges. In the third step, the gradient operator is applied a second time

$$[\mathcal{D}_1, \mathcal{D}_2] \begin{bmatrix} \mathcal{R} \cdot \mathcal{D}_1 \\ \mathcal{R} \cdot \mathcal{D}_2 \end{bmatrix} = \mathcal{D}_1(\mathcal{R} \cdot \mathcal{D}_1) + \mathcal{D}_2(\mathcal{R} \cdot \mathcal{D}_2). \tag{17.41}$$

Weickert [195] used a more sophisticated implicit solution scheme. However, the scheme is computationally more expensive and less isotropic than the explicit scheme in Eq. (17.41) if gradient operators are used that are optimized for isotropy as discussed in Section 12.5.5.

An even simpler but only approximate implementation of inhomogenous diffusion controls binomial smoothing using the operator

$$\mathcal{I} + \mathcal{R} \cdot (\mathcal{B} - \mathcal{I}). \tag{17.42}$$

The operator $\mathcal{R}$ computes the same scalar control image with values between zero and one as in Eq. (17.40).

Figure 17.5 shows the application of inhomogeneous diffusion for segmentation of noisy images. The test image contains a triangle and a rectangle. Standard smoothing significantly suppresses the noise but results in a significant blurring of the edges (Fig. 17.5b). Inhomogenous diffusion does not lead to a blurring of the edges and still results in a perfect segmentation of the square and the triangle (Fig. 17.5c). The only disadvantage is that the edges themselves remain noisy because no smoothing is performed at all there.

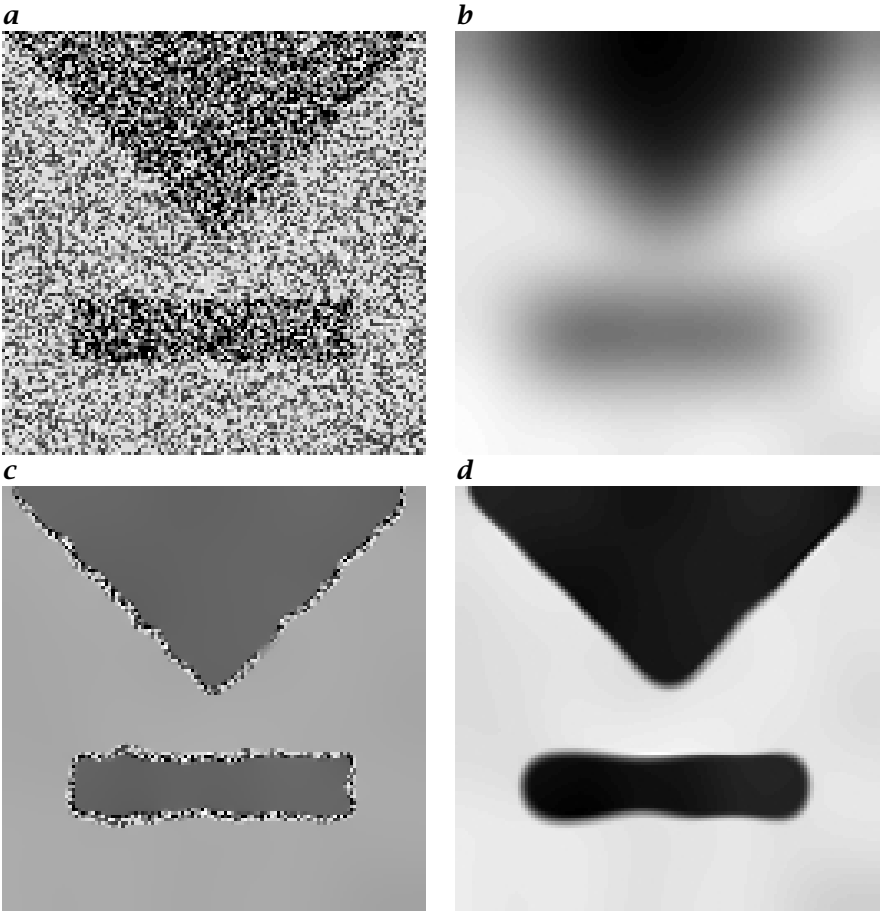

Figure 17.5: ***a*** *Original, smoothed by* ***b*** *linear diffusion,* ***c*** *inhomogenous but isotropic diffusion, and* ***d*** *anisotropic diffusion. From Weickert [195].*

17.5.2 Anisotropic Diffusion

As we have seen in the example discussed at the end of the last section, inhomogeneous diffusion has the significant disadvantage that it stops diffusion completely and in all directions at edges, leaving the edges noisy. However, edges are only blurred by diffusion perpendicular to them; diffusion parallel to them is even advantageous as it stabilizes the edges.

An approach that makes diffusion independent of the direction of edges is known as anisotropic diffusion. With this approach, the flux is no longer parallel to the gradient. Therefore, the diffusion can no longer be described by a scalar diffusion coefficient as in Eq. (5.7). Now,

a *diffusion tensor* is required:

$$\boldsymbol{j} = -\boldsymbol{D}\nabla f = -\begin{bmatrix} D_{11} & D_{12} \\ D_{12} & D_{22} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}. \tag{17.43}$$

With a diffusion tensor the diffusion-reaction system becomes

$$f_t = \nabla \left(\boldsymbol{D}(\nabla f \nabla f^T)\, \nabla f \right) - L_f \tag{17.44}$$

and the corresponding regularizer is

$$R = \operatorname{trace} \boldsymbol{R} \left(\nabla f \nabla f^T \right), \tag{17.45}$$

with $\boldsymbol{D} = \boldsymbol{R}'$. The properties of the diffusion tensor can best be seen if the symmetric tensor is brought into its principal-axis system by a rotation of the coordinate system. Then, Eq. (17.43) reduces to

$$\boldsymbol{j}' = -\begin{bmatrix} D_1' & 0 \\ 0 & D_2' \end{bmatrix} \begin{bmatrix} f_1' \\ f_2' \end{bmatrix} = -\begin{bmatrix} D_1' f_1' \\ D_2' f_2' \end{bmatrix}. \tag{17.46}$$

The diffusion in the two directions of the axes is now decoupled. The two coefficients on the diagonal, D_1' and D_2', are the *eigenvalues* of the diffusion tensor. By analogy to isotropic diffusion, the general solution for homogeneous anisotropic diffusion can be written as

$$f(\boldsymbol{x}, t) = \frac{1}{2\pi\sigma_1'(t)\sigma_2'(t)} \exp\left(-\frac{x'^2}{2\sigma_1'(t)} \right) * \exp\left(-\frac{y'^2}{2\sigma_2'(t)} \right) * f(\boldsymbol{x}, 0) \tag{17.47}$$

in the spatial domain with $\sigma_1'(t) = \sqrt{2D_1' t}$ and $\sigma_2'(t) = \sqrt{2D_2' t}$.

This means that anisotropic diffusion is equivalent to cascaded convolution with two 1-D Gaussian convolution kernels that are steered in the directions of the principal axes of the diffusion tensor. If one of the two eigenvalues of the diffusion tensor is significantly larger than the other, diffusion occurs only in the direction of the corresponding eigenvector. Thus the gray values are smoothed only in this direction. The spatial widening is — as for any diffusion process — proportional to the square root of the diffusion constant (Eq. (5.15)).

Using this feature of anisotropic diffusion, it is easy to design a diffusion process that predominantly smoothes only along edges but not perpendicularly to the edges. The orientation of the edges can be obtained, e. g., from the structure tensor (Section 13.3). With the following approach only smoothing across edges is hindered [195]:

$$\begin{aligned} D_1' &= 1 - \exp\left(-\frac{c_m}{(|\nabla(B^r * f)(\boldsymbol{x})|/\lambda)^m} \right) \\ D_2' &= 1. \end{aligned} \tag{17.48}$$

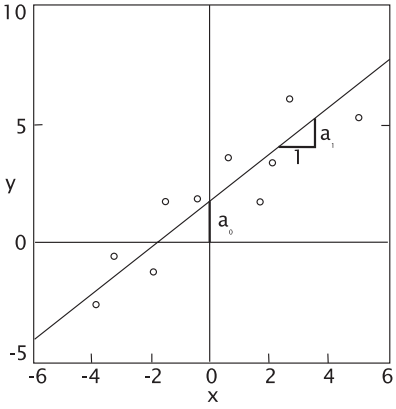

Figure 17.6: *Illustration of least-squares linear regression.*

As shown by Scharr and Weickert [162], an efficient and accurate explicit implementation of anisotropic diffusion is again possible with regularized first-order differential derivative optimized for minimum anisotropy:

$$[\mathcal{D}_1, \mathcal{D}_2] \begin{bmatrix} \mathcal{R}_{11} & \mathcal{R}_{12} \\ \mathcal{R}_{12} & \mathcal{R}_{22} \end{bmatrix} \begin{bmatrix} \mathcal{D}_1 \\ \mathcal{D}_2 \end{bmatrix} = \tag{17.49}$$
$$\mathcal{D}_1(\mathcal{R}_{11} \cdot \mathcal{D}_1 + \mathcal{R}_{12} \cdot \mathcal{D}_2) + \mathcal{D}_2(\mathcal{R}_{12} \cdot \mathcal{D}_1 + \mathcal{R}_{22} \cdot \mathcal{D}_2).$$

with

$$\begin{bmatrix} \mathcal{R}_{11} & \mathcal{R}_{12} \\ \mathcal{R}_{12} & \mathcal{R}_{22} \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \mathcal{D}_{x'} & 0 \\ 0 & \mathcal{D}_{y'} \end{bmatrix} \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}.$$

The $\mathcal{R}_{pq}$ are control images with values between zero and one that steer the diffusion into the direction parallel to edges at each point of the image.

Application of anisotropic diffusion shows that now — in contrast to inhomogeneous diffusion — the edges are also smoothed (Fig. 17.5d). The smoothing along edges has the disadvantage, however, that the corners of the edges are now blurred as with linear diffusion. This did not happen with inhomogeneous diffusion (Fig. 17.5c).

17.6 Discrete Inverse Problems†

In the second part of this chapter, we turn to discrete modeling. Discrete modeling can, of course, be derived, by directly discretizing the partial differential equations resulting from the variational approach.

Actually, we have already done this in Section 17.5 by the iterative discrete schemes for inhomogeneous and anisotropic diffusion.

However, by developing discrete modeling independently, we gain further insight. Again, we take another point of view of modeling and now regard it as a *linear discrete inverse problem*. As an introduction, we start with the familiar problem of linear regression and then develop the theory of discrete inverse modeling.

17.6.1 A Simple Example: Linear Regression

The fit of a straight line to a set of experimental data points x, y is a simple example of a discrete inverse problem. As illustrated in Fig. 17.6, the quantity y is measured as a function of a parameter x. In this case, our model is a straight line with two parameters, the offset a_0 and the slope a_1: $y = a_0 + a_1 x$. With a set of Q data points $[x_q, y_q]^T$ we end up with the linear equation system

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_Q \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_Q \end{bmatrix} \tag{17.50}$$

which can be abbreviated by

$$\boldsymbol{M}\boldsymbol{p} = \boldsymbol{d}. \tag{17.51}$$

The $Q \times 2$ matrix $\boldsymbol{M}$ is denoted as the *model matrix* or *design matrix*. This matrix reflects both the type of the model (here a linear regression) and the chosen independent measuring points x_q. The *model* or *parameter vector* $\boldsymbol{p}$ contains the parameters of the model to be estimated and the *data vector* $\boldsymbol{d}$ the measured data x_q.

If we have only two data points which do not coincide, $x_1 \neq x_2$, we get an exact solution of the linear equation system. If more than two data points are available, we have more equations than unknowns. We say that the equation system is an *overdetermined inverse problem*. In this case, it is generally no longer possible to obtain an exact solution. We can only compute an estimate of the model parameters $\boldsymbol{p}_{\text{est}}$ in the sense that the deviation of the data $\boldsymbol{d}$ from the data predicted with the model $\boldsymbol{d}_{\text{pre}} = \boldsymbol{M}\boldsymbol{p}_{\text{est}}$ is minimal. This deviation can be expressed by an *error vector* $\boldsymbol{e}$:

$$\boldsymbol{e} = \boldsymbol{d} - \boldsymbol{d}_{\text{pre}} = \boldsymbol{d} - \boldsymbol{M}\boldsymbol{p}_{\text{est}}. \tag{17.52}$$

17.6.2 Error Norms

In order to minimize the error vector we need a suitable measure. We may use *norms*, which we discussed when using inner product vector

spaces in Section 2.3.1. Generally, the L_n norm of the Q-dimensional vector $\boldsymbol{e}$ is defined as

$$\|\boldsymbol{e}\|_n = \left(\sum_{q=1}^{Q} |e_q|^n \right)^{1/n}. \tag{17.53}$$

A special case is the L_∞ norm

$$\|\boldsymbol{e}\|_\infty = \max_n |e_q|. \tag{17.54}$$

The L_2 norm is more commonly used; it is the root of the sum of the squared deviations of the error vector elements

$$\|\boldsymbol{e}\|_2 = \left(\sum_{q=1}^{Q} (d_q - d_{\text{pre},q})^2 \right)^{1/2}. \tag{17.55}$$

Higher norms rate higher deviations with a more significant weighting. The statistics of the data points determines which norm is to be taken. If the measured data points y_q have a *normal density* (Section 3.4.2), the L_2 norm must be used [124].

17.6.3 Least Squares Solution

The overdetermined linear inverse problem is solved with a minimum L_2 norm of the error vector by

$$\boldsymbol{p}_{\text{est}} = \left(\boldsymbol{M}^T \boldsymbol{M} \right)^{-1} \boldsymbol{M}^T \boldsymbol{d}. \tag{17.56}$$

This solution can be made plausible by the following sequence of operations:

$$\begin{array}{lcl|l} \boldsymbol{M}\boldsymbol{p}_{\text{est}} & = & \boldsymbol{d} & \boldsymbol{M}^T \\ \boldsymbol{M}^T\boldsymbol{M}\boldsymbol{p}_{\text{est}} & = & \boldsymbol{M}^T\boldsymbol{d} & \left(\boldsymbol{M}^T\boldsymbol{M}\right)^{-1} \\ \boldsymbol{p}_{\text{est}} & = & \left(\boldsymbol{M}^T\boldsymbol{M}\right)^{-1}\boldsymbol{M}^T\boldsymbol{d} & \end{array} \tag{17.57}$$

provided that the inverse of $\boldsymbol{M}^T\boldsymbol{M}$ exists.

17.6.4 Geometric Illustration

Before we study methods for solving huge linear equation systems, it is helpful to illustrate linear equation systems geometrically. The P model parameters $\boldsymbol{p}$ span a P"=dimensional vector space. This space can be regarded as the space of all possible solutions of an inverse problem with P model parameters. Now, we ask ourselves what it means to have

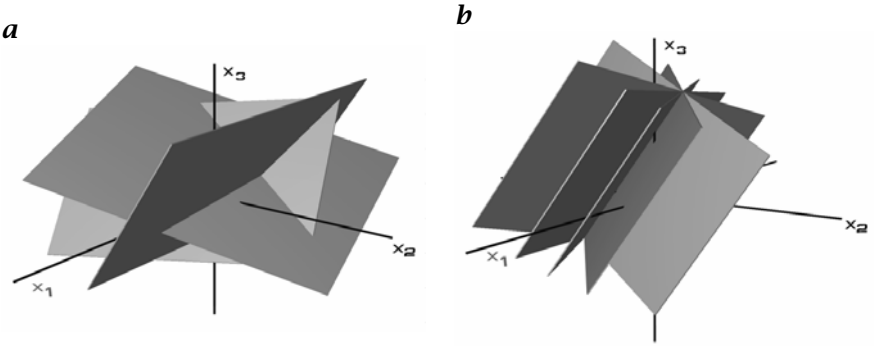

Figure 17.7: *Geometric illustration of the solution of a linear equation system with three unknowns using the Hough transform:* ***a*** *exact soluble equation system;* ***b*** *overdetermined equation system with a non-unique solution.*

one data point d_q. According to Eq. (17.51), one data point results in one linear equation involving all model parameters $\boldsymbol{M}$

$$\sum_{k=p'}^{P} m_{qp'} p_{p'} = d_q \quad \text{or} \quad \boldsymbol{g}_q \boldsymbol{p} = d_q. \tag{17.58}$$

This equation can be regarded as the scalar product of a row q of the model matrix $\boldsymbol{m}_q$ with the model vector $\boldsymbol{p}$. In the model space, this equation constitutes a $P-1$-dimensional *hyperplane* of all vectors $\boldsymbol{p}$ which has a normal vector $\boldsymbol{m}_q$ and a distance d_q from the origin of the *model space*. Thus, the linear equation establishes a one-to-one correspondence between a data point in the *data space* and a $(P-1)$-dimensional hyperplane in the model space. This mapping of data points into the model space is called the *Hough transform*, which we introduced in Section 16.5.2. Each data point reduces the space of possible solutions to a $(P-1)$-dimensional hyperplane in the model space.

Figure 17.7a illustrates the solution of a linear equation system with three unknowns. With three equations, three planes meet at a single point, provided that the corresponding 3×3 model matrix is invertible. Even in an overdetermined case, the solution needs not necessarily be unique. Figure 17.7b shows a case of five planes intersecting at a line. Then, the solution is not unique, but only restricted to a line. If this line is oriented along one of the axes, the corresponding model parameter may take any value; the two other model parameters, however, are fixed.

In case of an arbitrarily oriented line, things are more complex. Then, the parameter combinations normal to the line are fixed, but the parameter combination represented by a vector in the direction of the line is not. Using the *singular value decomposition* [54, 143], we can solve singular

linear equation systems and separate the solvable from the unsolvable parameter combinations.

An overdetermined linear equation system that has no unique solution is not just a mathematical curiosity. It is rather a common problem in image processing. We have encountered it already, for example in motion determination with the *aperture problem* (Section 14.3.2).

17.6.5 Derivation of the Least Squares Solution‡

In this section we provide a derivation of the solution of the overdetermined discrete linear inverse problem (Eq. (17.51)) that minimizes the L_2 norm of the error vector. Therefore, we compute the solution explicitly by minimizing the L_2 norm of the error vector $\boldsymbol{e}$ (Eq. (17.52)):

$$\|\boldsymbol{e}\|_2^2 = \sum_{q'=1}^{Q} \left(d_{q'} - \sum_{p'=1}^{P} m_{q'p'} p_{p'} \right) \left(d_{q'} - \sum_{p''=1}^{P} m_{q'p''} p_{p''} \right).$$

Factorizing the sum and interchanging the two summations yields

$$\begin{aligned} \|\boldsymbol{e}\|_2^2 &= \underbrace{\sum_{p'=1}^{P} \sum_{p''=1}^{P} p_{p'} p_{p''} \sum_{q'=1}^{Q} m_{q'p'} m_{q'p''}}_{A} \\ &- \underbrace{2 \sum_{p'=1}^{P} p_{p'} \sum_{q'=1}^{Q} m_{q'p'} d_{q'}}_{B} \\ &+ \sum_{q'=1}^{Q} d_{q'} d_{q'}. \end{aligned} \tag{17.59}$$

We find a minimum for this expression by computing the partial derivatives with respect to the parameters p_k that are to be optimized. Only the expressions A and B in Eq. (17.59) depend on p_k:

$$\begin{aligned} \frac{\partial A}{\partial p_k} &= \sum_{p'=1}^{P} \sum_{p''=1}^{P} \left(\delta_{k-p''} p_{p'} + \delta_{k-p'} p_{p''} \right) \sum_{q'=1}^{Q} m_{q'p'} m_{q'p''} \\ &= \sum_{p'=1}^{P} p_{p'} \sum_{q'=1}^{Q} m_{q'p'} m_{q'k} + \sum_{p''=1}^{P} p_{p''} \sum_{q'=1}^{Q} m_{q'k} m_{q'p''} \\ &= 2 \sum_{p'=1}^{P} m_{p'} \sum_{q'=1}^{Q} m_{q'p'} m_{q'k}, \\ \frac{\partial B}{\partial p_k} &= 2 \sum_{q'=1}^{Q} m_{q'k} d_{q'}. \end{aligned}$$

We add both derivatives and set them equal to zero:

$$\frac{\partial \|\boldsymbol{e}\|_2^2}{\partial p_k} = 2 \sum_{p'=1}^{P} p_{p'} \sum_{q'=1}^{Q} m_{q'k} m_{q'p'} - 2 \sum_{q'=1}^{Q} m_{q'k} d_{q'} = 0.$$

In order to express the sums as matrix-matrix and matrix-vector multiplications, we substitute the matrix $\boldsymbol{M}$ at two places by its transpose $\boldsymbol{M}^T$:

$$\sum_{p'=1}^{P} p_{p'} \sum_{q'=1}^{Q} m_{kq'}^T m_{q'p'} - \sum_{q'=1}^{Q} m_{kq'}^T d_{q'} = 0$$

and finally obtain the matrix equation

$$\underbrace{\underbrace{\underbrace{\boldsymbol{M}^T}_{P\times Q}\underbrace{\boldsymbol{M}}_{Q\times P}}_{P\times P}\underbrace{\boldsymbol{p}_{\text{est}}}_{P}}_{P} = \underbrace{\underbrace{\boldsymbol{M}^T}_{P\times Q}\underbrace{\boldsymbol{d}}_{Q}}_{P}. \tag{17.60}$$

This equation can be solved if the quadratic and symmetric $P \times P$ matrix $\boldsymbol{M}^T\boldsymbol{M}$ is invertible. Then

$$\boldsymbol{p}_{\text{est}} = \left(\boldsymbol{M}^T\boldsymbol{M}\right)^{-1}\boldsymbol{M}^T\boldsymbol{d}. \tag{17.61}$$

The matrix $(\boldsymbol{M}^T\boldsymbol{M})^{-1}\boldsymbol{M}^T$ is known as the *generalized inverse* $\boldsymbol{M}^{-g}$ of $\boldsymbol{M}$.

17.6.6 Error of Model Parameters

An overdetermined linear equation system that has been solved by minimizing the L_2 norm allows an analysis of the errors. We can study not only the deviations between model and data but also the errors of the etsimated model parameter vector $\boldsymbol{p}_{\text{est}}$.

The mean deviation between the measured and predicted data points is directly related to the norm of the error vector. The *variance* is

$$\sigma^2 = \frac{1}{Q-P}\|\boldsymbol{e}\|^2 = \frac{1}{Q-P}\|\boldsymbol{d} - \boldsymbol{M}\boldsymbol{p}_{\text{est}}\|_2^2. \tag{17.62}$$

In order not to introduce a bias in the estimate of the variance, we divide the norm by the *degree of freedom* $Q - P$ and not by Q.

According to Eq. (17.61), the estimated parameter vector $\boldsymbol{p}_{\text{est}}$ is a linear combination of the data vector $\boldsymbol{d}$. Therefore we can apply the *error propagation* law (Eq. (3.27)) derived in Section 3.3.3. The *covariance matrix* (for a definition see Eq. (3.19)) of the estimated parameter vector $\boldsymbol{p}_{\text{est}}$ using $(\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T$ is given by

$$\text{cov}(\boldsymbol{p}_{\text{est}}) = \left(\boldsymbol{M}^T\boldsymbol{M}\right)^{-1}\boldsymbol{M}^T\,\text{cov}(\boldsymbol{d})\,\boldsymbol{M}\left(\boldsymbol{M}^T\boldsymbol{M}\right)^{-1}. \tag{17.63}$$

If the individual elements in the data vector $\boldsymbol{d}$ are uncorrelated and have the same variance σ^2, i. e., $\text{cov}(\boldsymbol{d}) = \sigma^2\boldsymbol{I}$, Eq. (17.63) reduces to

$$\text{cov}(\boldsymbol{p}_{\text{est}}) = \left(\boldsymbol{M}^T\boldsymbol{M}\right)^{-1}\sigma^2. \tag{17.64}$$

In this case, $(\boldsymbol{M}^T\boldsymbol{M})^{-1}$ is — except for the factor σ^2 — directly the covariance matrix of the model parameters. This means that the diagonal elements contain the variances of the model parameters.

17.6.7 Regularization

So far, the error functional (Eq. (17.55)) only contains a similarity constraint but no regularization or smoothing constraint. For many discrete inverse problems — such as the linear regression discussed in Section 17.6.1 — a regularization of the parameters makes no sense. If the parameters to be estimated are, however, the elements of a time series or the pixels of an image, a smoothness constraint makes sense. A suitable smoothness parameter could then be the norm of the time series or image convolved by a derivative filter:

$$\|\boldsymbol{r}\|_2 = \|\boldsymbol{h} * \boldsymbol{p}\|_2^2 . \tag{17.65}$$

In the language of matrix algebra, convolution can be expressed by a vector matrix multiplication:

$$\|\boldsymbol{r}\|_2 = \|\boldsymbol{H}\boldsymbol{p}\|_2^2 . \tag{17.66}$$

Because of the convolution operation, the matrix $\boldsymbol{H}$ has a special form. Only the coefficients around the diagonal are nonzero, but all values in diagonal direction are the same.

As an example, we discuss the same smoothness criterion that we used also in the variational approach (Section 17.3.4), the first derivative. It can be approximated, for instance, by convolution with a forward difference filter that results into the matrix

$$\boldsymbol{H} = \begin{bmatrix} -1 & 1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 1 & 0 & \dots & 0 \\ 0 & 0 & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \end{bmatrix} . \tag{17.67}$$

Minimizing the combined error functional using the L_2 norm:

$$\|\boldsymbol{e}\|_2 = \underbrace{\|\boldsymbol{d} - \boldsymbol{M}\boldsymbol{p}\|_2^2}_{\text{similarity}} + \alpha^2 \underbrace{\|\boldsymbol{H}\boldsymbol{p}\|_2^2}_{\text{smoothness}} \tag{17.68}$$

results in the following least-squares solution [124]:

$$\boldsymbol{p}_{\text{est}} = \left(\boldsymbol{M}^T\boldsymbol{M} + \alpha^2\boldsymbol{H}^T\boldsymbol{H}\right)^{-1}\boldsymbol{M}^T\boldsymbol{d}. \tag{17.69}$$

The structure of the solution is similar to the least-squares solution in Eq. (17.56). The smoothness term just causes the additional term $\alpha^2\boldsymbol{H}^T\boldsymbol{H}$.

In the next section, we learn how to map an image to a vector, so that we can apply discrete inverse problems also to images.

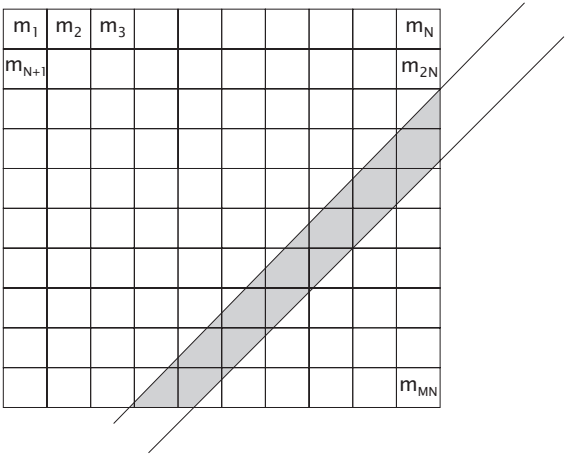

Figure 17.8: *Illustration of algebraic reconstruction from projections: a projection beam d_k crosses the image matrix. All the pixels met by the beam contribute to the projection.*

17.6.8 Algebraic Tomographic Reconstruction‡

In this section we discuss an example of a discrete inverse problem that includes image data: reconstruction from projections (Section 8.6). In order to apply the discrete inverse theory as discussed so far, the image data must be mapped onto a vector, the *image vector*. This mapping is easily performed by renumbering the pixels of the image matrix row by row (Fig. 17.8). In this way, an $M \times N$ image matrix is transformed into a column vector with the dimension $P = M \times N$:

$$\boldsymbol{p} = \left[m_1, m_2, \ldots, m_p, \ldots, m_P\right]^T. \tag{17.70}$$

Now we take a single projection beam that crosses the image matrix (Fig. 17.8). Then we can attribute a weighting factor to each pixel of the image vector that represents the contribution of the pixel to the projection beam. We can combine these factors in a Q-dimensional vector $\boldsymbol{g}_q$:

$$\boldsymbol{g}_q = \left[g_{q,1}, g_{q,2}, \ldots, g_{q,p}, \ldots, g_{Q,P}\right]^T. \tag{17.71}$$

The total emission or absorption along the qth projection beam d_q can then be expressed as the scalar product of the two vectors $\boldsymbol{g}_q$ and $\boldsymbol{p}$:

$$d_q = \sum_{p=1}^{P} g_{q,p} m_p = \boldsymbol{g}_q \boldsymbol{p}. \tag{17.72}$$

If Q projection beams cross the image matrix, we obtain a linear equation system of Q equations and P unknowns:

$$\underbrace{\boldsymbol{d}}_{Q} = \underbrace{\boldsymbol{M}}_{Q\times P}\underbrace{\boldsymbol{p}}_{P}. \tag{17.73}$$

The *data vector* $\boldsymbol{d}$ contains the measured projections and the *parameter vector* $\boldsymbol{p}$ contains the pixel values of the image matrix that are to be reconstructed. The *design matrix* $\boldsymbol{M}$ gives the relationship between these two vectors by describing how in a specific set up the projection beams cross the image matrix. With appropriate weighting factors, we can take into direct account the limited detector resolution and the size of the radiation source.

Algebraic tomographic reconstruction is a general and flexible method. In contrast to the filtered backprojection technique (Section 8.6.3) it is not limited to parallel projection. The beams can cross the image matrix in any manner and can even be curved. In addition, we obtain an estimate of the errors of the reconstruction.

However, algebraic reconstruction involves solving huge linear equation systems. At this point, it is helpful to illustrate the enormous size of these equation systems. In a typical problem, the model vector includes all pixels of an image. Even with moderate resolution, e. g., 256×256 pixels, the inverse of a 65536×65536 matrix would have to be computed. This matrix contains about $4 \cdot 10^9$ points and does not fit into the memory of any but the most powerful computers. Thus alternative solution techniques are required.

17.6.9 Further Examples of Inverse Problems

Problems of this kind are very common in the analysis of experimental data in natural sciences. Experimentalists look at a discrete inverse problem in the following way. They perform an experiment from which they gain a set of measuring results, and they combine them in a Q-dimensional data vector $\boldsymbol{d}$. These data are compared with a model of the observed process. The parameters of this model are given by a P-dimensional model vector $\boldsymbol{p}$. Now we assume that the relationship between the model and the data vector can be described as linear. It can then be expressed by a model matrix $\boldsymbol{M}$ and we obtain Eq. (17.73).

For image processing, inverse problems are also common. They do not only include the complete list of problems discussed in the introduction of this chapter (Section 17.1) but also optimization of filters. In this book, least-squares optimized filters for interpolation (Section 10.6.6) and edge detection (Sections 12.3.5 and 12.5.5) are discussed.

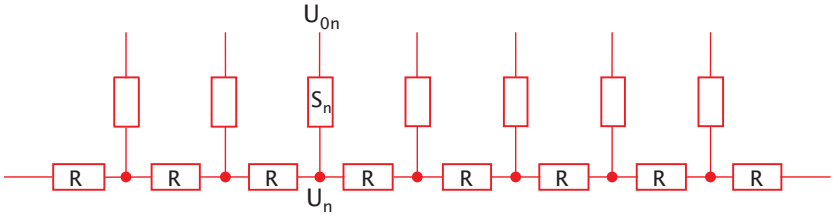

Figure 17.9: *Simple 1-D network for a 1-D smooth DVF; after Harris [62].*

17.7 Network Models‡

In this section we discuss another method emerging from electrical engineering, the *network model*. It has the advantage of being a discrete model which directly corresponds to discrete imagery. This section follows the work of Harris [61, 62]. The study of network models has become popular since network structures can be implemented directly on such massive parallel computer systems as the Connection Machine at Massachusetts Institute of Technology (MIT) [62] or in analog VLSI circuits [123].

17.7.1 One-Dimensional Networks‡

First, we consider the simple 1-D case. The displacement corresponds to an electric tension. Continuity is forced by interconnecting neighboring pixels with electrical resistors. In this way, we build up a linear resistor chain as shown in Fig. 17.9. We can force the displacement at a pixel to a certain value by applying a potential at the corresponding pixel. If only one voltage source exists in the resistor chain, the whole network is put to the same constant voltage. If another potential is applied to a second node of the network and all interconnecting resistors are equal, we obtain a linear voltage change between the two points. In summary, the network of resistors forces continuity in the voltage, while application of a voltage at a certain node forces similarity.

There are different types of boundary conditions. On the one hand, we can apply a certain voltage to the edge of the resistor chain and thus force a certain value of the displacement vector at the edge of the image. On the other hand, we can make no connection. This is equivalent to setting the first-order spatial derivative to zero at the edge. The voltage at the edge is then equal to the voltage at the next connection to a voltage source.

In the elasticity models (Section 17.3.5) we did not set the displacements to the value resulting from the similarity constraint directly, but allowed for some flexibility by applying the displacement via a spring. In a similar manner we apply the voltage, U_{0n}, to the node n not directly but via the resistor S_n (Fig. 17.9). We set the resistance proportional to the uncertainty of the displacement vector.

The difference equation for the network model is given by the rule that the sum of all currents must cancel each other at every node of the network. Using the definitions given in Fig. 17.9, we obtain for the node n of the network

$$\frac{U_n - U_{0n}}{S_n} + \frac{U_n - U_{n-1}}{R} + \frac{U_n - U_{n+1}}{R} = 0. \tag{17.74}$$

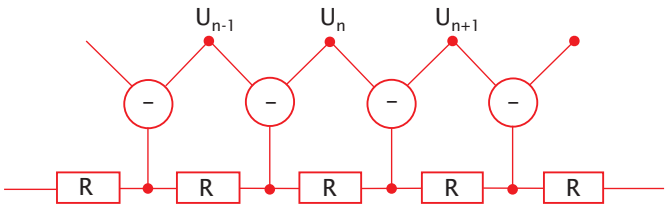

Figure 17.10: *Discrete network model for a 1-D scalar feature with smooth first-order derivatives; after Harris [62].*

The two fractions on the right side constitute the second-order discrete differentiation operator $\mathcal{D}_x^2$ (see Section 12.4.2). Thus Eq. (17.74) results in

$$\frac{1}{S}(U - U_0) - \frac{1}{R}\frac{\partial^2 U}{\partial x^2} = 0. \tag{17.75}$$

This equation is the 1-D form of Eq. (17.22). For a better comparison, we rewrite this equation for the 1-D case:

$$(\partial_x g)^2 \left(f + \frac{\partial_t g}{\partial_x g} \right) - \alpha^2 \frac{\partial^2 f}{\partial x^2} = 0. \tag{17.76}$$

Now we can quantify the analogy between the displacement vectors and the network model. The application of the potential U_0 corresponds to the computation of the local velocity by $-(\partial_t g)/(\partial_x g)$. The similarity and smoothness terms are weighted with the reciprocal resistance (conductance) $1/S$ and $1/R$ instead of with the squared gradient $(\partial_x g)^2$ and α^2.

17.7.2 Generalized Networks‡

Now we turn to the question of how to integrate the continuity of first-order derivatives into the network model. Harris [61] used an active subtraction module which computes the difference of two signals. All three connections of the element serve as both inputs and outputs. At two arbitrary inputs we apply a voltage and obtain the corresponding output voltage at the third connection.

Such a module requires active electronic components [61]. Figure 17.10 shows how this subtraction module is integrated into the network. It computes the difference voltage between two neighboring nodes. These differences — and not the voltages themselves — are put into the resistor network.

In this way we obtain a network that keeps the first derivative continuous. We can generalize this approach to obtain networks that keep higher-order derivatives continuous by adding several layers with subtraction modules (Fig. 17.11).

17.7.3 Discontinuities in Networks‡

Displacement vector fields show discontinuities at the edges of moving objects. Discontinuities can easily be implemented in the network model. In the simple network with zero-order continuity (Fig. 17.9), we just remove the connecting

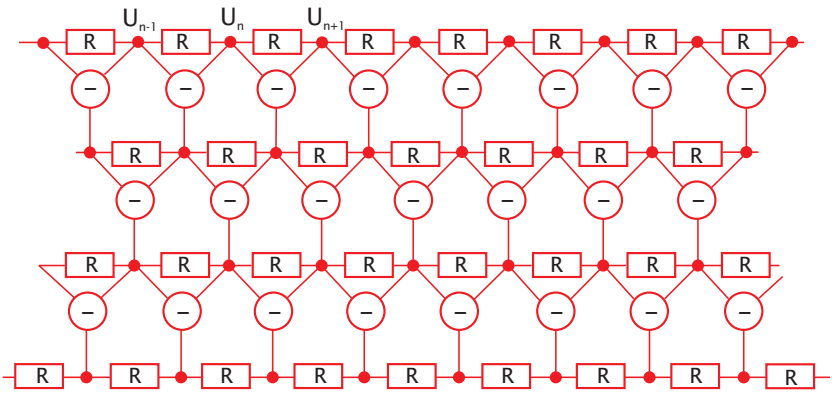

Figure 17.11: *Generalized network for a 1-D DVF that keeps higher-order derivatives smooth; after Harris [62].*

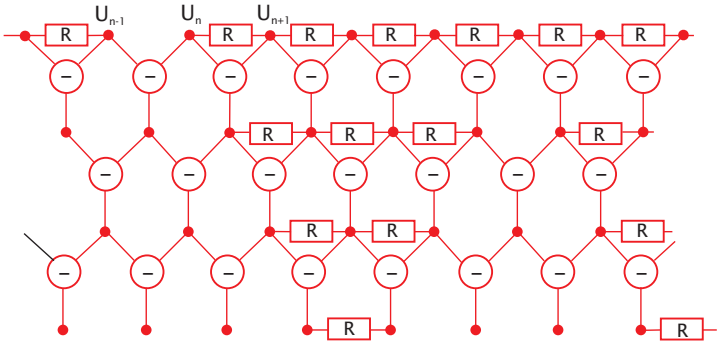

Figure 17.12: *Generalized 1-D network with a discontinuity in the DVF and its first spatial derivative as indicated.*

resistor between two neighboring nodes to produce a potential jump between these two nodes. In order to control the smoothness (Section 17.4), we can also think of a nonlinear network model with voltage-dependent resistors. We might suspect discontinuities at steep gradients in the velocity field. If the resistance increases with the tension, we have a mechanism to produce implied discontinuities. These brief considerations illustrate the flexibility and suggestiveness of network models.

Integration of discontinuities is more complex in a generalized network. Here we may place discontinuities at each level of the network, i. e., we may make either the DVF or any of its derivatives discontinuous by removing a resistor at the corresponding level. We need to remove all resistors of deeper-lying nodes that are connected to the point of discontinuity (Fig. 17.12). Otherwise, the higher-order derivatives stay continuous and cause the lower-order derivatives to become continuous.

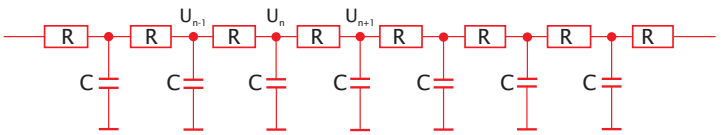

Figure 17.13: *1-D network with capacitors to simulate the convergence of iterative solutions.*

17.7.4 Two-Dimensional Networks‡

The network model can also be used for higher-dimensional problems. For a 2-D network model with zero-order continuity, we build up a 2-D mesh of resistors. The setup of generalized 2-D network models with higher-order continuity constraints is more complex. In each level we must consider the continuity of several partial derivatives. There are two first-order spatial derivatives, a horizontal and a vertical one. For each of them, we need to build up a separate layer with subtraction modules as shown in Fig. 17.10, in order to observe the smoothness constraint. Further details can be found in Harris [62].

17.7.5 Multigrid Networks‡

One of the most important practical issues is finding the rate of convergence of iterative methods for solving large equation systems in order to model them with networks. The question arises of whether it is also possible to integrate this important aspect into the network model. Iteration introduces a time dependency into the system, which can be modeled by adding capacitors to the network (Fig. 17.13). The capacitors do not change at all the static properties of the network.

When we start the iteration, we know the displacement vectors only at some isolated points. Therefore we want to know how many iterations it takes to carry this information to distant points where we do not have any displacement information. To answer this question, we derive the difference equation for the resistor-capacitor chain as shown in Fig. 17.13. It is given by the rule that the sum of all currents flowing into one node must be zero. In addition, we need to know that the current flowing into a capacitor is proportional to its capacitance C and the temporal derivative of the voltage $\partial U/\partial t$:

$$\frac{U_{n-1} - U_n}{R} + \frac{U_{n+1} - U_n}{R} - C\frac{\partial U_n}{\partial t} = 0 \tag{17.77}$$

or

$$\frac{\partial U_n}{\partial t} = \frac{(\Delta x)^2}{RC}\frac{\partial^2 U_n}{\partial x^2}. \tag{17.78}$$

In the second equation, we have introduced Δx as the spatial distance between neighboring nodes in order to formulate a spatial derivative. Also, $RC = \tau$, the time constant of an individual resistor-capacitor circuit. Equation (17.78) is the discrete 1-D formulation of one of the most important equations in natural sciences, the *transport* or *diffusion* equation, which we discussed in detail in Sections 5.2.1 and 17.5. Without explicitly solving Eq. (17.78), we can answer

the question as to the time constant needed to smooth the displacement vector field over a certain space scale. Let us assume a spatially varying potential with a wavelength λ decreasing exponentially with a time constant τ_λ that depends on the wavelength λ (compare Section 5.2.1):

$$U(x) = U_0(x)\exp(-t/\tau)\exp(\mathrm{i}kx). \tag{17.79}$$

Introducing this equation into Eq. (17.78), we obtain

$$\tau_\lambda = \frac{\tau}{(\Delta x\, k)^2} = \frac{\tau}{4\pi^2(\Delta x)^2}\lambda^2. \tag{17.80}$$

With this result, we can answer the question as to the convergence time of the iteration. The convergence time goes with the square of the wavelength of the structure. Consequently, it takes four times longer to get gray values at double the distance into equilibrium. Let us arbitrarily assume that we need one iteration step to bring neighboring nodes into equilibrium. We then need 100 iteration steps to equilibrate nodes that are 10 pixels distant. If the potential is only known at isolated points, this approach converges too slowly to be useful.

Multigrid data structures, which we discussed in Chapter 5, are an efficient tool to accelerate the convergence of the iteration. At the coarser levels of the pyramid, distant points come much closer together. In a pyramid with only six levels, the distances shrink by a factor of 32. Thus we can compute the large-scale structures of the DVF with a convergence rate that is about 1000 times faster than on the original image. We do not obtain any small-scale variations, but can use the coarse solution as the starting point for the iteration at the next finer level.

In this way, we can refine the solution from level to level and end up with a full-resolution solution at the lowest level of the pyramid. The computations at all the higher levels of the pyramid do not add a significant overhead, as the number of pixels at all levels of the pyramid is only one third more than at the lowest level. The computation of the DVF of the taxi scene (Fig. 17.3) with this method is shown in Fig. 17.4.

17.8 Inverse Filtering

In this section, we study a special class of inverse problems and show the way to fast iterative solutions of huge inverse problems.

17.8.1 Image Restoration

No image formation system is perfect because of inherent physical limitations. Therefore, images are not identical to their original. As scientific applications always push the limits, there is a need to correct for limitations in the sharpness of images. Humans also make errors in operating imaging systems. Images blurred by a misadjustment in the focus, smeared by the motion of objects or the camera or a mechanically unstable optical system, or degraded by faulty or misused optical systems are more common than we may think. A famous recent example

was the flaw in the optics of the Hubble space telescope where an error in the test procedures for the main mirror resulted in a significant residual aberration of the telescope. The correction of known and unknown image degradation is called *restoration*.

The question arises whether, and if so, to what extent, the effects of degradation can be reversed. It is obvious, of course, that information that is no longer present at all in the degraded image cannot be retrieved. To make this point clear, let us assume the extreme case that only the mean gray value of an image is retained. Then, it will not be possible by any means to reconstruct its content. However, images contain a lot of redundant information. Thus, we can hope that a distortion only partially removes the information of interest even if we can no longer "see" it directly.

In Sections 7.6 and 9.2.1, we saw that generally any optical system including digitization can be regarded as a *linear shift-invariant system* and, thus, described to a good approximation by a *point spread function* and a *transfer function*.

The first task is to determine and describe the image degradation as accurately as possible. This can be done by analyzing the image formation system either theoretically or experimentally by using some suitable test images. If this is not possible, the degraded image remains the only source of information.

17.8.2 Survey of Image Distortions

Given the enormous variety of ways to form images (Chapter 7), there are many reasons for image degradation. Imperfections of the optical system, known as lens aberrations, limit the sharpness of images. However, even with a perfect optical system, the sharpness is limited by diffraction of electromagnetic waves at the aperture stop of the lens. While these types of degradation are an inherent property of a given optical system, blurring by *defocusing* is a common misadjustment that limits the sharpness in images. Further reasons for blurring in images are unwanted motions and vibrations of the camera system during the exposure time. Especially systems with a narrow field of view (telelenses) are very sensitive to this kind of image degradation. Blurring can also occur when objects move more than a pixel at the image plane during the exposure time.

Defocusing and *lens aberrations* are discussed together in this section as they are directly related to the optical system. The effect of blurring or aberration is expressed by the *point spread function* $h(\boldsymbol{x})$ and the *optical transfer function* (*OTF*); see Section 7.6. Thus, the relation between object $g(\boldsymbol{x})$ and image $g'(\boldsymbol{x})$ is in the spatial and Fourier domain

$$g'(\boldsymbol{x}) = (h * g)(\boldsymbol{x}) \quad \circ\!\!-\!\!\bullet \quad \hat{g}'(\boldsymbol{k}) = \hat{h}(\boldsymbol{k})\hat{g}(\boldsymbol{k}). \tag{17.81}$$

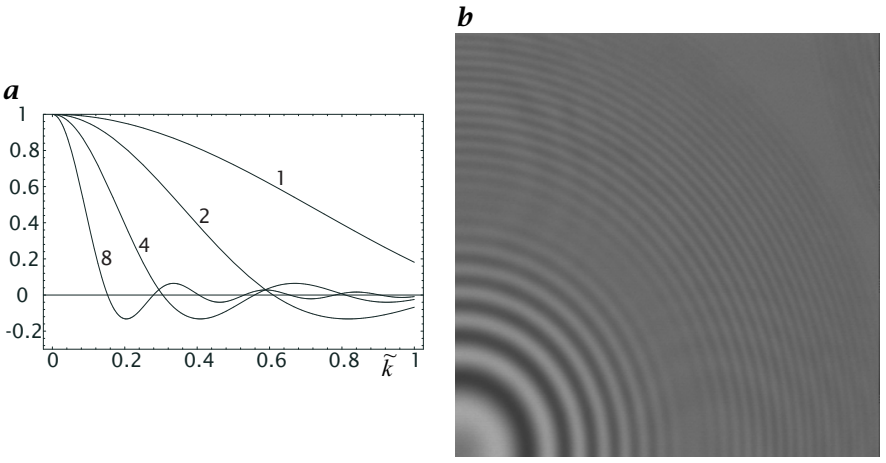

Figure 17.14: ***a*** *Transfer functions for disk-shaped blurring. The parameters for the different curves are the radius of the blur disk;* ***b*** *defocused image of the ring test pattern.*

Lens aberrations are generally more difficult to handle. Most aberrations increase strongly with distance from the optical axis and are, thus, not shift invariant and cannot be described with a position-independent PSF. However, the aberrations change only slowly and continuously with the position in the image. As long as the resulting blurring is limited to an area in which we can consider the aberration to be constant, we can still treat them with the theory of linear shift-invariant systems. The only difference is that the PSF and OTF vary gradually with position.

If defocusing is the dominant blurring effect, the PSF has the shape of the aperture stop. As most aperture stops can be approximated by a circle, the function is a disk. The Fourier transform of a disk with radius r is a Bessel function of the form (≻ R5):

$$\frac{1}{\pi r^2}\Pi\left(\frac{|\boldsymbol{x}|}{2r}\right) \circ\!\!-\!\!\bullet \frac{2J_1(|\boldsymbol{k}|r)}{|\boldsymbol{k}|r}. \tag{17.82}$$

This Bessel function, as shown in Fig. 17.14a, has a series of zeroes and, thus, completely eliminates certain wave numbers. This effect can be observed in Fig. 17.14b, which shows a defocused image of the ring test pattern.

While blurring by defocusing and lens aberrations tend to be isotropic, blurring effects by motion are one-dimensional, as shown in Fig. 17.15b. In the simplest case, motion is constant during the exposure. Then, the PSF of motion blur is a one-dimensional box function. Without loss of generality, we first assume that the direction of motion is along the x

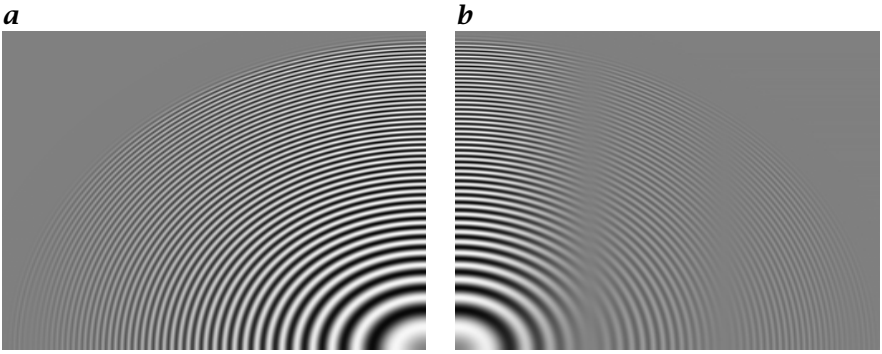

Figure 17.15: *Simulation of blurring by motion using the ring test pattern:* ***a*** *small and* ***b*** *large velocity blurring in horizontal direction.*

axis. Then,

$$h_{Bl}(x) = \frac{1}{2u\Delta t}\Pi\left(\frac{x}{2u\Delta t}\right) \circ\!\!-\!\!\bullet\, \hat{h}_{Bl}(k) = \frac{\sin(ku\Delta t/2)}{ku\Delta t/2}, \tag{17.83}$$

where u is the magnitude of the velocity and Δt the exposure time. The blur length is $\Delta x = u\Delta t$.

If the velocity $\boldsymbol{u}$ is oriented in another direction, Eq. (17.83) can be generalized to

$$h_{Bl}(\boldsymbol{x}) = \frac{1}{2|\boldsymbol{u}|\Delta t}\Pi\left(\frac{\boldsymbol{x}\bar{\boldsymbol{u}}}{1|\boldsymbol{u}|\Delta t}\right)\delta(\boldsymbol{u}\boldsymbol{x}) \circ\!\!-\!\!\bullet\, \hat{h}_{Bl}(\boldsymbol{k}) = \frac{\sin(\boldsymbol{k}\boldsymbol{u}\Delta t/2)}{\boldsymbol{k}\boldsymbol{u}\Delta t/2}, \tag{17.84}$$

where $\bar{\boldsymbol{u}} = \boldsymbol{u}/|\boldsymbol{u}|$ is a unit vector in the direction of the motion blur.

17.8.3 Deconvolution

Common to defocusing, motion blur, and 3-D imaging by such techniques as focus series or confocal microscopy (Section 8.2.4) is that the object function $g(\boldsymbol{x})$ is convolved by a point spread function. Therefore, the principal procedure for reconstructing or restoring the object function is the same. Essentially, it is a *deconvolution* or an *inverse filtering* as the effect of the convolution by the PSF is to be inverted. Given the simple relations in Eq. (17.81), inverse filtering is in principle an easy procedure. The effect of the convolution operator $\mathcal{H}$ is reversed by the application of the inverse operator $\mathcal{H}^{-1}$. In the Fourier space we can write:

$$\hat{\boldsymbol{G}}_R = \frac{\hat{\boldsymbol{G}}'}{\hat{\boldsymbol{H}}'} = \hat{\boldsymbol{H}}^{-1} \cdot \hat{\boldsymbol{G}}'. \tag{17.85}$$

The reconstructed image $\boldsymbol{G}_R$ is then given by applying the inverse Fourier transform:

$$\boldsymbol{G}_R = \mathcal{F}^{-1}\hat{\boldsymbol{H}}^{-1} \cdot \mathcal{F}\hat{\boldsymbol{G}}'. \tag{17.86}$$

The reconstruction procedure is as follows. The Fourier transformed image, $\mathcal{F}\boldsymbol{G}'$, is multiplied by the inverse of the OTF, $\hat{\boldsymbol{H}}^{-1}$, and then transformed back to the spatial domain. The inverse filtering can also be performed in the spatial domain by convolution with a mask that is given by the inverse Fourier transform of the inverse OTF:

$$\boldsymbol{G}_R = (\mathcal{F}^{-1}\hat{\boldsymbol{H}}^{-1}) * \boldsymbol{G}'. \tag{17.87}$$

At first glance, inverse filtering appears straightforward. In most cases, however, it is useless or even impossible to apply Eqs. (17.86) and (17.87). The reason for the failure is related to the fact that the OTF is often zero in wide ranges. The OTFs for motion blur (Eq. (17.84)) and defocusing (Eq. (17.82)) have extended zero range. In these areas, the inverse OTF becomes infinite.

Not only the zeroes of the OTF cause problems; already all the ranges in which the OTF becomes small do so. This effect is related to the influence of noise. For a quantitative analysis, we assume the following simple image formation model:

$$\boldsymbol{G}' = \boldsymbol{H} * \boldsymbol{G} + \boldsymbol{N} \quad \circ\!\!-\!\!\bullet \quad \hat{\boldsymbol{G}}' = \hat{\boldsymbol{H}} \cdot \hat{\boldsymbol{G}} + \hat{\boldsymbol{N}} \tag{17.88}$$

Equation (17.88) states that the noise is added to the image *after* the image is degraded. With this model, according to Eq. (17.85), inverse filtering yields

$$\hat{\boldsymbol{G}}_R = \hat{\boldsymbol{H}}^{-1} \cdot \hat{\boldsymbol{G}}' = \hat{\boldsymbol{G}} + \hat{\boldsymbol{H}}^{-1} \cdot \hat{\boldsymbol{N}} \tag{17.89}$$

provided that $\hat{\boldsymbol{H}} \neq 0$. This equation states that the restored image is the restored original image $\hat{\boldsymbol{G}}$ plus the noise amplified by $\hat{\boldsymbol{H}}^{-1}$.

If $\hat{\boldsymbol{H}}$ tends to zero, $\hat{\boldsymbol{H}}^{-1}$ becomes infinite, and so does the noise level. Equations (17.88) and (17.89) also state that the signal to noise ratio is not improved at all but remains the same because the noise and the useful image content in the image are multiplied by the same factor.

From this basic fact we can conclude that inverse filtering does not improve the image quality at all. More generally, it is clear that no linear technique will do so. All we can do with linear techniques is to amplify the structures attenuated by the degradation up to the point where the noise level still does not reach a critical level.

As an example, we discuss the 3-D reconstruction from microscopic *focus series*. A focus series is an image stack of microscopic images in which we scan the focused depth. Because of the limited *depth of field* (Section 7.4.3), only objects in a thin plane are imaged sharply. Therefore, we obtain a 3-D image. However, it is distorted by the point spread function of optical imaging. Certain structures are completely filtered out and blurred objects are superimposed over sharply imaged objects. We can now use inverse filtering to try to limit these distortions.

It is obvious that an exact knowledge of the PSF is essential for a good reconstruction. In Section 7.6.1, we computed the 3-D *PSF* of optical imaging neglecting lens errors and resolution limitation due to diffraction. However, high magnification microscopy images are diffraction-limited.

The diffraction-limited 3-D PSF was computed by Erhardt et al. [35]. The resolution limit basically changes the double cone of the 3-D PSF (Fig. 7.13) only close to the focal plane. At the focal plane, a point is no longer imaged to a point but to a diffraction disk. As a result, the *OTF* drops off to higher wave numbers in the $k_x k_y$ plane. To a first approximation, we can regard the diffraction-limited resolution as an additional lowpass filter by which the OTF is multiplied for geometrical imaging and by which the PSF is convolved.

The simplest approach to obtain an optimal reconstruction is to limit application of the inverse OTF to the wave number components that are not damped below a critical threshold. This threshold depends on the noise in the images. In this way, the true inverse OTF is replaced by an *effective inverse OTF* which approaches zero again in the wave number regions that cannot be reconstructed.

The result of such a reconstruction procedure is shown in Fig. 17.16. A $64 \times 64 \times 64$ focus series has been taken of the nucleus of a cancerous rat liver cell. The resolution in all directions is $0.22\,\mu$m. The images clearly verify the theoretical considerations. The reconstruction considerably improves the resolution in the xy image plane, while the resolution in the z direction — as expected — is clearly worse. Structures that change in the z direction are completely eliminated in the focus series by convolution with the PSF of optical images and therefore can not be reconstructed.

17.8.4 Iterative Inverse Filtering

Iterative techniques form an interesting variant of inverse filtering as they give control over the degree of reconstruction to be applied. Let $\mathcal{H}$ be the blurring operator. We introduce the new operator $\mathcal{H}' = \mathcal{I} - \mathcal{H}$. Then the inverse operator

$$\mathcal{H}^{-1} = \frac{\mathcal{I}}{\mathcal{I} - \mathcal{H}'} \tag{17.90}$$

can be approximated by the Taylor expansion

$$\mathcal{H}^{-1} = \mathcal{I} + \mathcal{H}' + \mathcal{H}'^2 + \mathcal{H}'^3 + \ldots, \tag{17.91}$$

or, written explicitly for the OTF in the continuous Fourier domain,

$$\hat{h}^{-1}(\boldsymbol{k}) = 1 + \hat{h}' + \hat{h}'^2 + \hat{h}'^3 + \ldots. \tag{17.92}$$

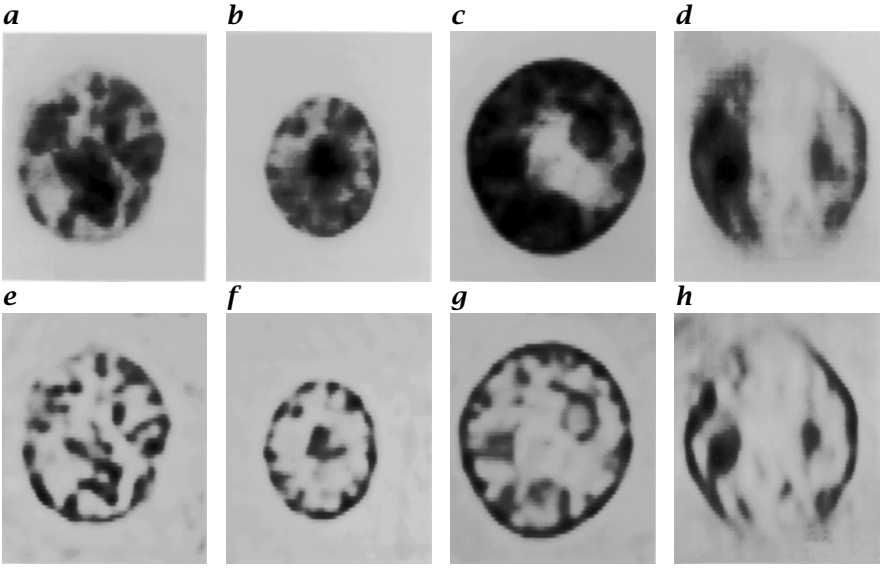

Figure 17.16: *3-D reconstruction of a focus series of a cell nucleus taken with conventional microscopy. Upper row:* ***a–c*** *selected original images;* ***d*** xz *cross section perpendicular to the image plane. Lower row:* ***e–h*** *reconstructions of the images* ***a–d*** *; courtesy of Dr. Schmitt and Prof. Dr. Komitowski, German Cancer Research Center, Heidelberg.*

In order to understand how the iteration works, we consider periodic structures. First, we take one that is only slightly attenuated. This means that $\hat{h}$ is only slightly less than one. Thus, $\hat{h}'$ is small and the iteration converges rapidly.

The other extreme is when the periodic structure has nearly vanished. Then, $\hat{h}'$ is close to one. Consequently, the amplitude of the periodic structure increases by the same amount with each iteration step (linear convergence). This procedure has the significant advantage that we can stop the iteration as soon as the noise patterns become noticeable.

A direct application of the iteration makes not much sense because the increasing exponents of the convolution masks become larger and thus the computational effort increases from step to step. A more efficient scheme known as *Van Cittert iteration* utilizes Horner's scheme for polynomial computation:

$$\boldsymbol{G}_0 = \boldsymbol{G}', \quad \boldsymbol{G}_{k+1} = \boldsymbol{G}' + (\boldsymbol{I} - \boldsymbol{H}) * \boldsymbol{G}_k. \tag{17.93}$$

In Fourier space, it is easy to examine the convergence of this iteration. From Eq. (17.93)

$$\hat{g}_k(\boldsymbol{k}) = \hat{g}'(\boldsymbol{k}) \sum_{i=0}^{k} (1 - \hat{h}(\boldsymbol{k}))^i. \tag{17.94}$$

This equation constitutes a geometric series with the start value $a_0 = \hat{g}'$ and the factor $q = 1 - \hat{h}$. The series converges only if $|q| = |1 - \hat{h}| < 1$. Then the sum is given by

$$\hat{g}_k(\boldsymbol{k}) = a_0 \frac{1 - q^k}{1 - q} = \hat{g}'(\boldsymbol{k}) \frac{1 - |1 - \hat{h}(\boldsymbol{k})|^k}{\hat{h}(\boldsymbol{k})} \tag{17.95}$$

and converges to the correct value $\hat{g}'/\hat{h}$. Unfortunately, this condition for convergence is not met for all transfer functions that have negative values. Therefore the Van Cittert iteration cannot be applied to motion blurring and to defocusing.

A slight modification of the iteration process, however, makes it possible to use it also for degradations with partially negative transfer functions. The simple trick is to apply the transfer function twice. The transfer function $\hat{h}^2$ of the cascaded filter $\boldsymbol{H} * \boldsymbol{H}$ is positive.

The modified iteration scheme is

$$\boldsymbol{G}_0 = \boldsymbol{H} * \boldsymbol{G}', \quad \boldsymbol{G}_{k+1} = \boldsymbol{H} * \boldsymbol{G}' + (\boldsymbol{I} - \boldsymbol{H} * \boldsymbol{H}) * \boldsymbol{G}_k. \tag{17.96}$$

With $a_0 = \hat{h}\hat{g}'$ and $q = 1 - \hat{h}^2$ the iteration again converges to the correct value

$$\lim_{k \to \infty} \hat{g}_k(\boldsymbol{k}) = \lim_{k \to \infty} \hat{h}\hat{g}' \frac{1 - |1 - \hat{h}^2|^k}{\hat{h}^2} = \frac{\hat{g}'}{\hat{h}}, \quad \text{if} \quad |1 - \hat{h}^2| < 1 \tag{17.97}$$

17.9 Further Readings‡

This subject of this chapter relies heavily on matrix algebra. Golub and van Loan [54] give an excellent survey on matrix computations. Variational methods (Section 17.3) are expounded by Jähne et al. [83, Vol. 2, Chapter 16] and Schnörr and Weickert [164]. The usage of the membran model (Section 17.3.5) was first reported by Broit [12], who applied it in computer tomography. Later it was used and extended by Dengler [29] for image sequence processing. Nowadays, elasticity models are a widely used tool in quite different areas of image processing such as modeling and tracking of edges [91], reconstruction of 3-D objects [183] and reconstruction of surfaces [182]. Anisotropic diffusion (Section 17.5) and nonlinear scale spaces are an ongoing research topic. An excellent account is given by Weickert [195] and Jähne et al. [83, Vol. 2, Chapter 15]. Optimal filters for fast anisotropic diffusion are discussed by Scharr and Weickert [162] and Scharr and Uttenweiler [161].

18 Morphology

18.1 Introduction

In Chapters 16 and 17 we discussed the segmentation process that extracts objects from images, i. e., identifies which pixels belong to which objects. Now we can perform the next step and analyze the *shape* of the objects. In this chapter, we discuss a class of neighborhood operations on binary images, the morphological operators that modify and analyze the form of objects.

18.2 Neighborhood Operations on Binary Images

18.2.1 Binary Convolution

In our survey of digital image processing, operators relating pixels in a small neighborhood emerged as a versatile and powerful tool for scalar and vector images (Chapter 4). The result of such an operation in binary images can only be a zero or a one. Consequently, neighborhood operators for binary images will work on the shape of object, adding pixels to an object or deleting pixels from an object. In Sections 4.2 and 4.4 we discussed the two basic operations for combining neighboring pixels of gray value images: convolution ("weighting and summing up") and rank value filtering ("sorting and selecting"). With binary images, we do not have much choice as to which kind of operations to perform. We can combine pixels only with the logical operations of Boolean algebra. We might introduce a *binary convolution* by replacing the multiplication of the image and mask pixels with an *and operation* and the summation by an *or operation*:

$$g'_{mn} = \bigvee_{m'=-R}^{R} \bigvee_{n'=-R}^{R} m_{m',n'} \wedge g_{m+m',n+n'}. \tag{18.1}$$

The $\wedge$ and $\vee$ denote the logical *and* and *or* operations, respectively. The binary image $\boldsymbol{G}$ is convolved with a symmetric $2R+1 \times 2R+1$ mask $\boldsymbol{M}$. Note that in contrast to convolution operations, the mask is not mirrored at the origin (see Section 4.2.5).

What does this operation achieve? Let us assume that all the coefficients of the mask are set to 'one'. If one or more object pixels, i. e.,

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2

Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

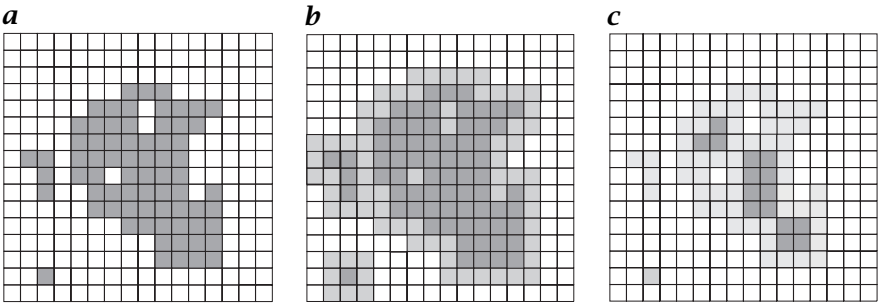

Figure 18.1: ***b*** *Dilation and* ***c*** *erosion of a binary object in* ***a*** *with a* 3×3 *mask. The removed (erosion) and added (dilation) pixels are shown in a lighter color.*

'ones', are within the mask, the result of the operation will be one, otherwise it is zero (Fig. 18.1a, b). Hence, the object will be dilated. Small holes or cracks will be filled and the contour line will become smoother, as shown in Fig. 18.2b. The operator defined by Eq. (18.1) is known as the *dilation operator*.

Interestingly, we can end up with the same effect if we apply *rank-value filter* (see Section 4.4) to binary images. Let us take the *maximum operator*. The maximum will then be one if one or more 'ones' are within the mask, just as with the binary convolution operation in Eq. (18.1).

The *minimum operator* has the opposite effect. Now the result is only one if the mask is completely within the object (Fig. 18.1c). In this way the object is eroded. Objects smaller than the mask disappear completely and objects connected only by a small bridge will become disconnected. The *erosion* of an object can also be performed using binary convolution with logical *and* operations:

$$g'_{mn} = \bigwedge_{m'=-R}^{R} \bigwedge_{n'=-R}^{R} m_{m',n'} \wedge g_{m+m',n+n'} \tag{18.2}$$

For higher-dimensional images, Eqs. (18.1) and (18.2) just need to be appended by another loop for each coordinate. In 3-D space, the dilation operator is, for instance,

$$g'_{lmn} = \bigvee_{l'=-R}^{R} \bigvee_{m'=-R}^{R} \bigvee_{n'=-R}^{R} m_{l'm'n'} \wedge g_{l+l',m+m',n+n'}. \tag{18.3}$$

By transferring the concepts of neighborhood operations for gray value images to binary images we have gained an important tool to operate on the form of objects. We have already seen in Fig. 18.1 that these operations can be used to fill small holes and cracks or to eliminate small objects.

The size of the mask governs the effect of the operators, therefore the mask is often called the *structure element*. For example, an erosion operation works like a net that has holes in the shape of the mask. All objects that fit through the hole will slip through and disappear from the image. An object remains only if at least at one point the mask is completely covered by object pixels. Otherwise it disappears.

An operator that works on the form of objects is called a *morphological operator*. The name originates from the research area of morphology which describes the form of objects in biology and geosciences.

18.2.2 Operations on Sets

We used a rather unconventional way to introduce morphological operations. Normally, these operations are defined as operations on sets of pixels. We regard $\boldsymbol{G}$ as the set of all the pixels of the matrix that are not zero. $\boldsymbol{M}$ is the set of the non-zero mask pixels. With $\boldsymbol{M}_p$ we denote the mask shifted with its reference point (generally but not necessarily its center) to the pixel p. Erosion is then defined as

$$\boldsymbol{G} \ominus \boldsymbol{M} = \{p : \boldsymbol{M}_p \subseteq \boldsymbol{G}\} \tag{18.4}$$

and dilation as

$$\boldsymbol{G} \oplus \boldsymbol{M} = \{p : \boldsymbol{M}_p \cap \boldsymbol{G} \neq \emptyset\}. \tag{18.5}$$

These definitions are equivalent to Eqs. (18.1) and (18.2), respectively. We can now express the erosion of the set of pixels $\boldsymbol{G}$ by the set of pixels $\boldsymbol{M}$ as the set of all the pixels p for which $\boldsymbol{M}_p$ is completely contained in $\boldsymbol{G}$. In contrast, the dilation of $\boldsymbol{G}$ by $\boldsymbol{M}$ is the set of all the pixels for which the intersection between $\boldsymbol{G}$ and $\boldsymbol{M}_p$ is not an empty set. As the set-theoretical approach leads to more compact and illustrative formulas, we will use it from now on.

Equations Eqs. (18.1) and (18.2) still remain important for the implementation of morphological operations with logical operations. The erosion and dilation operators can be regarded as elementary morphological operators from which other more complex operators can be built. Their properties are studied in detail in the next section.

18.3 General Properties

Morphological operators share most but not all of the properties we have discussed for linear convolution operators in Section 4.2. The properties discussed below are not restricted to 2-D images but are generally valid for N-dimensional image data.

18.3.1 Shift Invariance

Shift invariance results directly from the definition of the erosion and dilation operator as convolutions with binary data in Eqs. (18.1) and (18.2). Using the *shift operator* S as defined in Eq. (4.17) and the operator notation, we can write the shift invariance of any morphological operator $\mathcal{M}$ as

$$\mathcal{M}\,({}^{mn}S\boldsymbol{G}) = {}^{mn}S\,(\mathcal{M}\boldsymbol{G})\,. \tag{18.6}$$

18.3.2 Principle of Superposition

What does the *superposition principle* for binary data mean? For gray value images it is defined as

$$\mathcal{H}(a\boldsymbol{G} + b\boldsymbol{G}') = a\mathcal{H}\boldsymbol{G} + b\mathcal{H}\boldsymbol{G}'. \tag{18.7}$$

The factors a and b make no sense for binary images; the addition of images corresponds to the union or *logical or* of images. If the superposition principle is valid for morphological operations on binary images, it has the form

$$\mathcal{M}(\boldsymbol{G} \cup \boldsymbol{G}') = (\mathcal{M}\boldsymbol{G}) \cup (\mathcal{M}\boldsymbol{G}') \quad \text{or} \quad \mathcal{M}(\boldsymbol{G} \vee \boldsymbol{G}') = (\mathcal{M}\boldsymbol{G}) \vee (\mathcal{M}\boldsymbol{G}'). \tag{18.8}$$

The operation $\boldsymbol{G} \vee \boldsymbol{G}'$ means a pointwise *logical or* of the elements of the matrices $\boldsymbol{G}$ and $\boldsymbol{G}'$. Generally, morphological operators are not additive in the sense of Eq. (18.8). While the dilation operation conforms to the superposition principle, erosion does not. The erosion of the union of two objects is generally a superset of the union of two eroded objects:

$$\begin{array}{lcl} (\boldsymbol{G} \cup \boldsymbol{G}') \ominus \boldsymbol{M} & \supseteq & (\boldsymbol{G} \ominus \boldsymbol{M}) \cup (\boldsymbol{G}' \ominus \boldsymbol{M}) \\ (\boldsymbol{G} \cup \boldsymbol{G}') \oplus \boldsymbol{M} & = & (\boldsymbol{G} \oplus \boldsymbol{M}) \cup (\boldsymbol{G}' \oplus \boldsymbol{M}). \end{array} \tag{18.9}$$

18.3.3 Commutativity and Associativity

Morphological operators are generally not *commutative*:

$$\boldsymbol{M}_1 \oplus \boldsymbol{M}_2 = \boldsymbol{M}_2 \oplus \boldsymbol{M}_1, \text{ but } \boldsymbol{M}_1 \ominus \boldsymbol{M}_2 \neq \boldsymbol{M}_2 \ominus \boldsymbol{M}_1. \tag{18.10}$$

We can see that erosion is not commutative if we take the special case that $\boldsymbol{M}_1 \supset \boldsymbol{M}_2$. Then the erosion of $\boldsymbol{M}_2$ by $\boldsymbol{M}_1$ yields the empty set. However, both erosion and dilation masks consecutively applied in a cascade to the same image $\boldsymbol{G}$ are commutative:

$$\begin{array}{lll} (\boldsymbol{G} \ominus \boldsymbol{M}_1) \ominus \boldsymbol{M}_2 & = G \ominus (\boldsymbol{M}_1 \oplus \boldsymbol{M}_2) & = (\boldsymbol{G} \ominus \boldsymbol{M}_2) \ominus \boldsymbol{M}_1 \\ (\boldsymbol{G} \oplus \boldsymbol{M}_1) \oplus \boldsymbol{M}_2 & = \boldsymbol{G} \oplus (\boldsymbol{M}_1 \oplus \boldsymbol{M}_2) & = (\boldsymbol{G} \oplus \boldsymbol{M}_2) \oplus \boldsymbol{M}_1. \end{array} \tag{18.11}$$

These equations are important for the implementation of morphological operations. Generally, the cascade operation with k structure elements

$\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots, \boldsymbol{M}_k$ is equivalent to the operation with the structure element $\boldsymbol{M} = \boldsymbol{M}_1 \oplus \boldsymbol{M}_2 \oplus \ldots \oplus \boldsymbol{M}_k$. In conclusion, we can decompose large structure elements in the very same way as we decomposed linear shift-invariant operators. An important example is the composition of separable structure elements by the horizontal and vertical elements $\boldsymbol{M} = \boldsymbol{M}_x \oplus \boldsymbol{M}_y$. Another less trivial example is the construction of large one-dimensional structure elements from structure elements including many zeros:

$$\begin{array}{c} [1\ 1\ 1] \oplus [1\ 0\ 1] = [1\ 1\ 1\ 1\ 1] \\ [1\ 1\ 1\ 1\ 1] \oplus [1\ 0\ 0\ 0\ 1] = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1] \\ [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1] \oplus [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1] \\ = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1] \end{array} . \tag{18.12}$$

In this way, we can build up large exponentially growing structure elements with a minimum number of logical operations just as we built up large convolution masks by cascading in Section 11.6. It is more difficult to obtain isotropic, i. e., circular-shaped, structure elements. The problem is that the dilation of horizontal and vertical structure elements always results in a rectangular-shaped structure element, but not in a circular mask. A circular mask can be approximated, however, with one-dimensional structure elements running in more directions than only along the axes. As with smoothing convolution masks, large structure elements can be built efficiently by cascading multistep masks.

18.3.4 Monotony

Erosion and dilation are monotonic operations

$$\begin{array}{lcl} \boldsymbol{G}_1 \subseteq \boldsymbol{G}_2 & \rightsquigarrow & \boldsymbol{G}_1 \oplus \boldsymbol{M} \subseteq \boldsymbol{G}_2 \oplus \boldsymbol{M} \\ \boldsymbol{G}_1 \subseteq \boldsymbol{G}_2 & \rightsquigarrow & \boldsymbol{G}_1 \ominus \boldsymbol{M} \subseteq \boldsymbol{G}_2 \ominus \boldsymbol{M}. \end{array} \tag{18.13}$$

Monotony means that the subset relations are invariant with respect to erosion and dilation.

18.3.5 Distributivity

Linear shift-invariant operators are *distributive* with regard to addition. The corresponding distributivities for erosion and dilation with respect to the union and intersection of two images $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ are more complex:

$$\begin{array}{lcl} (\boldsymbol{G}_1 \cap \boldsymbol{G}_2) \oplus \boldsymbol{M} & \subseteq & (\boldsymbol{G}_1 \oplus \boldsymbol{M}) \cap (\boldsymbol{G}_2 \oplus \boldsymbol{M}) \\ (\boldsymbol{G}_1 \cap \boldsymbol{G}_2) \ominus \boldsymbol{M} & = & (\boldsymbol{G}_1 \ominus \boldsymbol{M}) \cap (\boldsymbol{G}_2 \ominus \boldsymbol{M}) \end{array} \tag{18.14}$$

and

$$\begin{array}{lcl} (\boldsymbol{G}_1 \cup \boldsymbol{G}_2) \oplus \boldsymbol{M} & = & (\boldsymbol{G}_1 \oplus \boldsymbol{M}) \cup (\boldsymbol{G}_2 \oplus \boldsymbol{M}) \\ (\boldsymbol{G}_1 \cup \boldsymbol{G}_2) \ominus \boldsymbol{M} & \supseteq & (\boldsymbol{G}_1 \ominus \boldsymbol{M}) \cup (\boldsymbol{G}_2 \ominus \boldsymbol{M}). \end{array} \tag{18.15}$$

Erosion is distributive over the intersection operation, while dilation is distributive over the union operation.

18.3.6 Duality

Erosion and dilation are *dual operators*. By negating the binary image, erosion is converted to dilation and vice versa:

$$\begin{aligned} \overline{G} \ominus M &= \overline{G \oplus M} \\ \overline{G} \oplus M &= \overline{G \ominus M}. \end{aligned} \tag{18.16}$$

The mathematical foundation of morphological operations including complete proofs for all the properties stated in this section can be found in the classic book by Serra [168].

18.4 Composite Morphological Operators

18.4.1 Opening and Closing

Using the elementary erosion and dilation operations we now develop further useful operations to work on the form of objects. While in the previous section we focused on the general and theoretical aspects of morphological operations, we now concentrate on application.

The erosion operation is useful for removing small objects. However, it has the disadvantage that all the remaining objects shrink in size. We can avoid this effect by dilating the image after erosion with the same structure element. This combination of operations is called an *opening operation*

$$G \circ M = (G \ominus M) \oplus M. \tag{18.17}$$

The opening sieves out all objects which at no point completely contain the structure element, but avoids a general shrinking of object size (Fig. 18.2c, d). It is also an ideal operation for removing lines with a thickness smaller than the diameter of the structure element. Note also that the object boundaries become smoother.

In contrast, the dilation operator enlarges objects and closes small holes and cracks. General enlargement of objects by the size of the structure element can be reversed by a following erosion (Fig. 18.3d and e). This combination of operations is called a *closing operation*

$$G \bullet M = (G \oplus M) \ominus M. \tag{18.18}$$

The area change of objects with different operations may be summarized by the following relations:

$$G \ominus M \subseteq G \circ M \subseteq G \subseteq G \bullet M \subseteq G \oplus M. \tag{18.19}$$

Opening and closing are *idempotent operations*:

$$\begin{aligned} G \bullet M &= (G \bullet M) \bullet M \\ G \circ M &= (G \circ M) \circ M, \end{aligned} \tag{18.20}$$

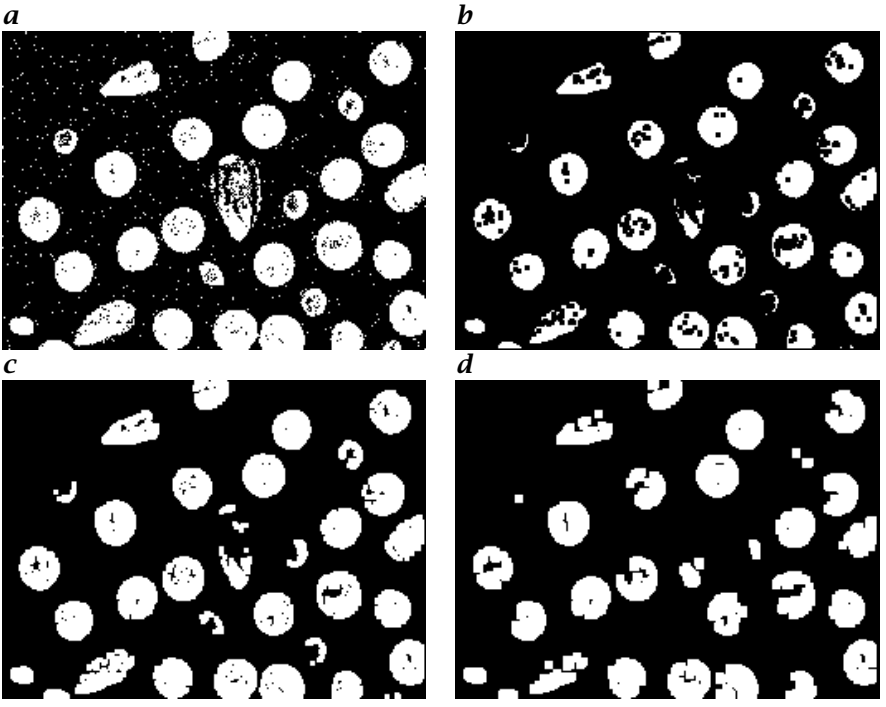

Figure 18.2: *Erosion and opening:* ***a*** *original binary image;* ***b*** *erosion with a* 3×3 *mask;* ***c*** *opening with a* 3×3 *mask;* ***d*** *opening with a* 5×5 *mask.*

i. e., a second application of a closing and opening with the same structure element does not show any further effects.

18.4.2 Hit-Miss Operator

The *hit-miss operator* originates from the question of whether it is possible to detect objects of a specific shape. The erosion operator only removes objects that at no point completely contain the structure element and thus removes objects of very different shapes. Detection of a specific shape requires a combination of two morphological operators. As an example, we discuss the detection of objects containing horizontal rows of three consecutive pixels.

If we erode the image with a 1×3 mask that corresponds to the shape of the object

$$\boldsymbol{M}_1 = [1\ 1\ 1]\,, \tag{18.21}$$

we will remove all objects that are smaller than the target object but retain also all objects that are larger than the mask, i. e., where the shifted mask is a subset of the object $\boldsymbol{G}$ ($(\boldsymbol{M}_p \subseteq \boldsymbol{G})$, Fig. 18.4d). Thus, we now

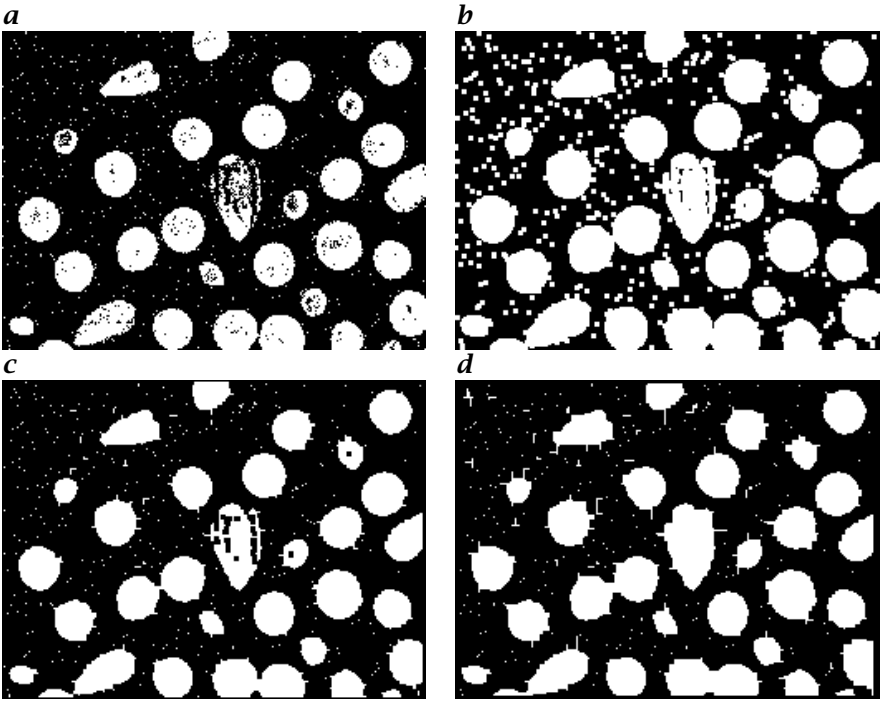

Figure 18.3: *Dilation and closing:* ***a*** *original binary image;* ***b*** *dilation with a* 3×3 *mask;* ***c*** *closing with a* 3×3 *mask;* ***d*** *closing with a* 5×5 *mask.*

need a second operation to remove all objects larger than the target object.

This can be done by analyzing the background of the original binary image. Thus, we can use as a second step an erosion of the background with a 3×5 mask $\boldsymbol{M}_2$ in which all coefficients are zero except for the pixels in the background, surrounding the object. This is a negative mask for the object:

$$\boldsymbol{M}_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{18.22}$$

The eroded background then contains all pixels in which the background has the shape of $\boldsymbol{M}_2$ or larger ($\boldsymbol{M}_2 \subseteq \overline{\boldsymbol{G}}$, Fig. 18.4b). This corresponds now to objects having the sought shape or a smaller one. As the first erosion obtains all objects equal to or larger than the target, the intersection of the image eroded with $\boldsymbol{M}_1$ and the background eroded with $\boldsymbol{M}_2$ gives all center pixels of the objects with horizontal rows of three consecutive pixels (Fig. 18.4e). In general, the *hit-miss operator* is

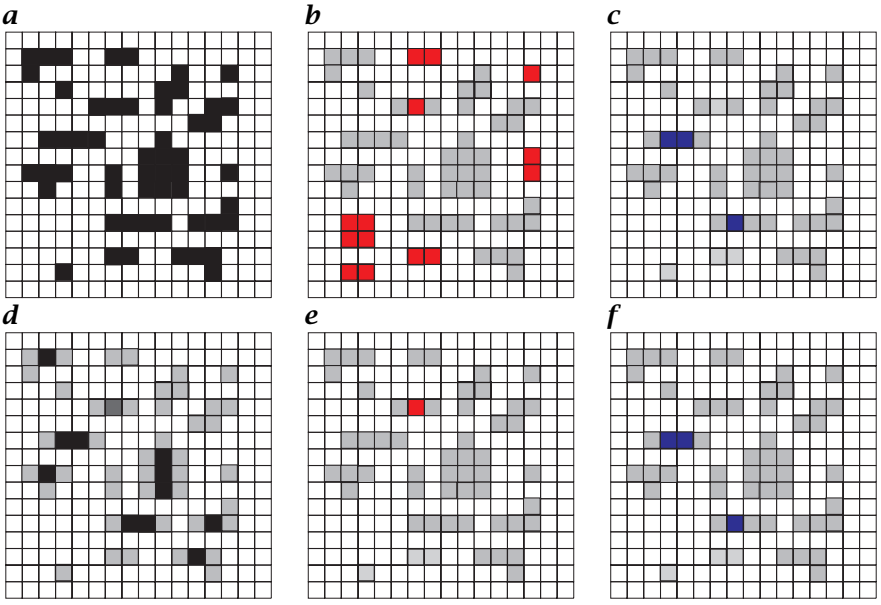

Figure 18.4: *Illustration of the hit-miss operator for extracting all objects containing horizontal rows of three consecutive pixels:* ***a*** *original image;* ***b*** *background eroded by a* 3×5 *mask (Eq. (18.22), negative mask of the object);* ***c*** *background eroded by the* 3×7 *mask (Eq. (18.24));* ***d*** *object eroded by the* 1×3 *mask (Eq. (18.21));* ***e*** *intersection of* ***b*** *and* ***d*** *extracting the objects with horizontal rows of 3 consecutive pixels;* ***f*** *intersection of* ***c*** *and* ***d*** *extracting objects with 3 to 5 horizontal rows of consecutive pixels in a* 3×7 *free background.*

defined as

$$\begin{aligned} \boldsymbol{G} \otimes (\boldsymbol{M}_1, \boldsymbol{M}_2) &= (\boldsymbol{G} \ominus \boldsymbol{M}_1) \cap (\overline{\boldsymbol{G}} \ominus \boldsymbol{M}_2) \\ &= (\boldsymbol{G} \ominus \boldsymbol{M}_1) \cap \overline{(\boldsymbol{G} \oplus \boldsymbol{M}_2)} \end{aligned} \tag{18.23}$$

with the condition that $\boldsymbol{M}_1 \cap \boldsymbol{M}_2 = \emptyset$, because otherwise the hit-miss operator would result in the empty set.

With the hit-miss operator, we have a flexible tool with which we can detect objects of a given specific shape. The versatility of the hit-miss operator can easily be demonstrated by using another miss mask

$$\boldsymbol{M}_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{18.24}$$

Erosion of the background with this mask leaves all pixels in the binary image where the union of the mask $\boldsymbol{M}_3$ with the object is zero (Fig. 18.4c). This can only be the case for objects with horizontal rows of one to five

consecutive pixels in a 3×7 large background. Thus, the hit-miss operator with $\boldsymbol{M}_1$ and $\boldsymbol{M}_3$ gives all center pixels of objects with horizontal rows of 3 to 5 consecutive pixels in a 3×7 large background (Fig. 18.4f).

As the hit and miss masks of the hit-miss operator are disjunct, they can be combined into one mask using a hit (1), miss (0), and don't care (x) notation. The combined mask is marked by 1 where the hit mask is one, by 0 where the miss mask is one, and by x where both masks are zero. Thus, the hit-miss mask for detecting objects with horizontal rows of 3 to 5 consecutive pixels is

$$\boldsymbol{M} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x & 1 & 1 & 1 & x & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{18.25}$$

If a hit-miss mask has no don't-care pixels, it extracts objects of an exact shape given by the 1-pixels of the mask. If don't-care pixels are present in the hit-miss mask, the 1-pixels give the minimum and the union of the 1-pixels and don't-care pixels the maximum of the detected objects.

As another example, the hit-mass mask

$$\boldsymbol{M}_I = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{18.26}$$

detects isolated pixels. Thus, the operation $\boldsymbol{G} / \boldsymbol{G} \otimes \boldsymbol{M}_I$ removes isolated pixels from a binary image. The / symbol represents the set difference operator.

The hit-miss operator detects certain shapes only if the miss mask surrounds the hit mask. If the hit mask touches the edge of the hit-miss mask, only certain shapes at the border of an object are detected. The hit-miss mask

$$\boldsymbol{M}_C = \begin{bmatrix} x & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{18.27}$$

for instance, detects lower right corners of objects.

18.4.3 Thinning

Often it is required to apply an operator that erodes an object but does not break it apart into different pieces. With such an operator, the topology of the object is preserved and line-like structures can be reduced to a one-pixel thickness. Unfortunately, the erosion operator does not have this feature. It can break an object into pieces.

A proper thinning operator, however, must not erode a point under the following conditions: (i) an object must not break into two pieces, (ii) an end point must not be removed so that the object does not become shorter, and (iii) an object must not be deleted. We illustrate this approach with the thinning of an object that has an 8-neighborhood connectivity. This corresponds to an erosion by a mask with 4-neighborhood connectivity:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

The mask contains five nonzero points. Thus there are $2^5 = 32$ possible binary patterns to which the thinning operator can be applied. Normal erosion would give a nonzero result only when all five points are set. A thinning operator will not erode a point under the following additional conditions:
(i) do not break object into two pieces:

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

(ii) do not remove end point:

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

and (iii) do not delete an object:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Thus only in 8 out of 16 cases that contain a nonzero central point, it is removed. Because there are only 32 possible patterns, the thinning operator can be implemented in an efficient way by a 32-entry look-up table with binary output. The addresses for the table are computed in the following way. Each point in the mask corresponds to a bit of the address that is set if the pixel of the binary image on that position is set.

The thinning operator is an iterative procedure which can be repeated until no further changes occur. Therefore it makes sense to use a second look-up table that gives a one if the thinning operator results in a change. In this way, it is easy to count changes and detect when no more changes occur.

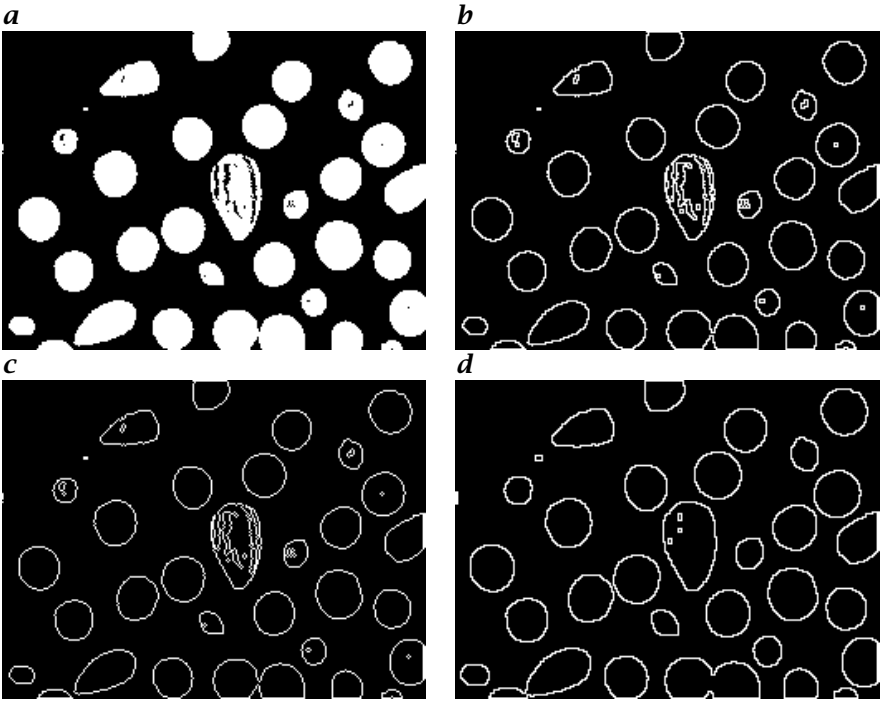

Figure 18.5: *Boundary extraction with morphological operators:* ***a*** *original binary image;* ***b*** *8-connected and* ***c*** *4-connected boundaries extracted with* $\boldsymbol{M}_{b4}$ *and* $\boldsymbol{M}_{b8}$*, respectively, Eq. (18.28);* ***d*** *8-connected boundary of the background extracted by using Eq. (18.30).*

18.4.4 Boundary Extraction

Morphological operators can also be used to extract the boundary of a binary object. This operation is significant as the boundary is a complete yet compact representation of the geometry of an object from which further shape parameters can be extracted, as we discuss later in this chapter.

Boundary points miss at least one of their neighbors. Thus, an erosion operator with a mask containing all possible neighbors removes all boundary points. These masks for the 4- and 8-neighborhood are:

$$\boldsymbol{M}_{b4} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{M}_{b8} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{18.28}$$

The boundary is then obtained by the set difference (/ operator) between the object and the eroded object:

$$\begin{aligned} \partial G &= G/(G \ominus M_b) \\ &= G \cap \overline{(G \ominus M_b)} \\ &= G \cap (\bar{G} \oplus M_b). \end{aligned} \tag{18.29}$$

As Eq. (18.29) shows, the boundary is also given as the intersection of the object with the dilated background. Figure 18.5 shows 4- and 8-connected boundaries extracted from binary objects using Eq. (18.28).

The boundary of the background is similarly given by dilating the object and subtracting it:

$$\partial G_B = (G \oplus M_b)/G. \tag{18.30}$$

18.4.5 Distance transforms

The boundary consists of all points with a distance zero to the edge of the object. If we apply boundary extraction again to the object eroded with the mask Eq. (18.28), we obtain all points with a distance of one to the boundary of the object. A recursive application of the boundary extraction procedure thus gives the distance of all points of the object to the boundary. Such a transform is called a *distance transform* and can be written as

$$D = \bigcup_{n=1}^{\infty} \left[(G \ominus M_b^{n-1})/(G \ominus M_b^n) \cdot n \right], \tag{18.31}$$

where the operation $\cdot$ denotes pointwise multiplication of the binary image of the nth distance contour with the number n.

This straightforward distance transform has two serious flaws. First, it is a slow iterative procedure. Second, it does not give the preferred Euclidian distance but — depending on the chosen neighborhood connectivity — the city block or chess board distance (Section 2.2.3).

Fortunately, fast algorithms are available for computing the Euclidian distance. The Euclidian distance transform is an important transform because it introduces isotropy for morphological operations. All morphological operations suffer from the fact that the Euclidian distance is not a natural measure on a rectangular grid. Square-shaped structure elements, for instance, all inherit the chess board distance. Successive dilation with such structure elements makes the objects look more and more like squares, for instance.

The Euclidian distance transform can be used to perform isotropic erosion and dilation operations. For an erosion operation with a radius r, we keep only pixels with a distance greater than r in the object. In

a similar way, an isotropic dilation can be performed by computing a Euclidian distance transform of the background and then an isotropic erosion of the background.

18.5 Further Readings‡

The authoritative source for the theory of morphological image processing is a monograph written by the founders of image morphology, see Serra [168]. The more practical aspects are covered by Jähne and Haußecker [82, Chapter 14] and Soille [175]. Meanwhile morphological image processing is a mature research area with a solid theoretical foundation and a wide range of applications as can be seen from recent conference proceeding, e. g., Serra and Soille [169].

19 Shape Presentation and Analysis

19.1 Introduction

All operations discussed in Chapters 11–15 extracted features from images that are represented as images again. Even the morphological operations discussed in Chapter 18 that analyze and modify the shape of segmented objects in binary images work in this way. It is obvious, however, that the shape of objects can be represented in a much more compact form. All information on the shape of an object is, for example, contained in its boundary pixels.

In Section 19.2, we therefore address the question of how to represent a segmented object. We will study the representation of binary objects with the *run-length code* (Section 19.2.1), the *quadtree* (Section 19.2.2), and the *chain code* (Section 19.2.3). Two further object representations, *moments* and *Fourier descriptors*, are of such significance that we devote entire sections to them (Sections 19.3 and 19.4).

A compact representation for the shape of objects is not of much use if it takes a lot of effort to compute it and if it is cumbersome to compute shape parameters directly from it. Therefore we address also the question of shape parameter extraction from the different shape representations in Section 19.5.

Shape parameters are extracted from objects in order to describe their shape, to compare it to the shape of template objects, or to partition objects into classes of different shapes. In this respect the important questions arises how shape parameters can be made invariant on certain transformations. Objects can be viewed from different distances and from different points of view. Thus it is of interest to find shape parameters that are scale and rotation invariant or that are even invariant under affine or perspective projection.

19.2 Representation of Shape

19.2.1 Run-Length Code

A compact, simple, and widely used representation of an image is *run-length code*. The run-length code is produced by the following procedure. An image is scanned line by line. If a line contains a sequence of p equal

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2

Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

a) ***Gray value image***

Original line (hex): 12 12 12 20 20 20 20 25 27 25 20 20 20 20 20 20

Code (hex): 82 **12** 83 **20** 2 **25 27 25** 85 **20**

b) ***Binary image***

Original line (hex): 1 1 1 1 1 1 0 0 0 1 1 1 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1

Code (hex) 0 6 3 3 2 1 5 8

Figure 19.1: *Demonstration of the run-length code for* ***a*** *gray value image,* ***b*** *binary image.*

pixels, we do not store p times the same value, but store the pixel value once and indicate that it occurs p times (Fig. 19.1). In this way, large uniform line segments can be stored very efficiently.

For binary images, the code can be especially efficient as only the two pixel values zero and one occur. As a sequence of zeroes is always followed by a sequence of ones, there is no need to store the pixel value. We only need to store the number of times a pixel value occurs (Fig. 19.1b). We must be careful at the beginning of a line, however, as it may begin with a one or a zero. This problem can be resolved if we assume a line to begin with zero. If a line should start with a sequence of ones, we start the run-length code with a zero to indicate that the line begins with a sequence of zero zeroes (Fig. 19.1b).

Run-length code is suitable for compact storage of images. It has become an integral part of several standard image formats, for example, the TGA or the *TIFF* file formats. Run-length code is less useful for direct processing of images, however, because it is not object oriented. As a result, run-length encoding is more useful for compact image storage. Not all types of images can be successfully compressed with this scheme. Digitized gray-scale images, for example, always contain some noise so that the probability for sufficiently long sequences of pixels with the same gray value is very low. However, high data reduction factors can be achieved with binary images and many types of computer-generated gray-scale and color images.

19.2.2 Quadtrees

The run-length codes discussed in Section 19.2.1 are a line-oriented representation of binary images. Thus they encode one-dimensional rather than two-dimensional data. The two-dimensional structure is actually not considered at all. In contrast, a *quadtree* is based on the principle of recursive decomposition of space, as illustrated in Fig. 19.2 for a binary image.

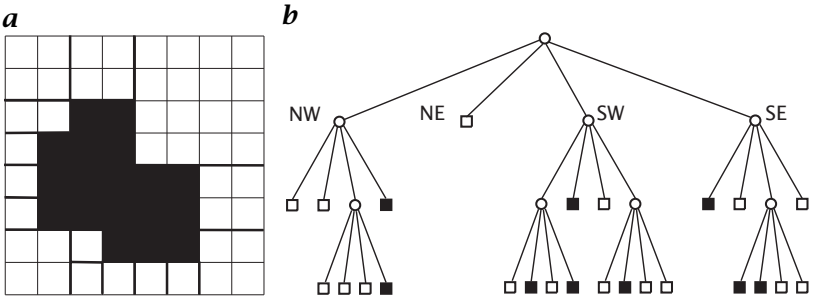

Figure 19.2: *Representation of a binary image by a region quadtree:* ***a*** *successive subdivision of the binary image into quadrants;* ***b*** *the corresponding region quadtree.*

First, the whole image is decomposed into four equal-sized *quadrants*. If one of the quadrants does not contain a uniform region, i. e., the quadrant is not included entirely in the object or background, it is again subdivided into four subquadrants. The decomposition stops if only uniform quadrants are encountered or if the quadrants finally contain only one pixel.

The recursive decomposition can be represented in a data structure known in computer science as *tree* (Fig. 19.2b). At the top level of the tree, the *root*, the decomposition starts. The root corresponds to the entire binary image. It is connected via four edges to four *child nodes* which represent from left to right the NW, NE, SW, and SE quadrants. If a quadrant needs no further subdivision, it is represented by a *terminal node* or a *leaf node* in the tree. It is called black when the quadrant belongs to an object and white otherwise, indicated by a filled and open square, respectively. Nonleaf nodes require further subdivision and are said to be gray and shown as open circles (Fig. 19.2b).

Quadtrees can be encoded, for example, by a *depth-first traversal* of the tree starting at the root. It is only required to store the type of the node with the symbols b (black), w (white), and g (gray). We start the code with the value of the root node. Then we list the values of the child nodes from left to right. Each time we encounter a gray node, we continue encoding at one level lower in the tree. This rule is applied recursively. This means that we return to a higher level in the tree only after the visited branch is completely encoded down to the lowest level. This is why this encoding is named depth-first.

The example quadtree shown in Fig. 19.2b results in the code

$$ggwwgwwwbbwggwbwbbwgwbwwgbwgbbwww.$$

The code becomes more readable if we include a left parenthesis each time we descend one level in the tree and a right parenthesis when we

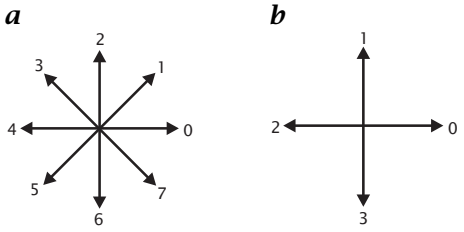

Figure 19.3: *Direction coding in* ***a*** *an 8-neighborhood and* ***b*** *a 4-neighborhood.*

ascend again

$$g(g(wwg(wwwb)b)wg(g(wbwb)bwg(wbww))g(bwg(bbww)w)).$$

However, the code is unique without the parentheses. A quadtree is a compact representation of a binary image if it contains many leaf nodes at high levels. However, in the worst case, for example a regular checkerboard pattern, all leaf nodes are at the lowest level. The quadtree then contains as many leaf nodes as pixels and thus requires many more bytes of storage space than the direct representation of the binary image as a matrix.

The region quadtree discussed here is only one of the many possibilities for recursive spatial decomposition. Three-dimensional binary images can be recursively decomposed in a similar way. The 3-D image is subdivided into eight equally sized octants. The resulting data structure is called a region *octree*. Quadtrees and octrees have gained significant importance in geographic information systems and computer graphics.

Quadtrees are a more suitable encoding technique for images than the line-oriented run-length code. But they are less suitable for image analysis. It is rather difficult to perform shape analysis directly on quadtrees. Without going into further details, this can be seen from the simple fact that an object shifted by one pixel in any direction results in a completely different quadtree. Region quadtrees share their most important disadvantage with run-length code: the technique is a global image decomposition and not one that represents objects extracted from images in a compact way.

19.2.3 Chain Code

In contrast to run-length code and quadtrees, *chain code* is an object-related data structure for representing the boundary of a binary object effectively on a discrete grid. Instead of storing the positions of all the boundary pixels, we select a starting pixel and store only its coordinate. If we use an algorithm that scans the image line by line, this will be the uppermost left pixel of the object. Then we follow the boundary

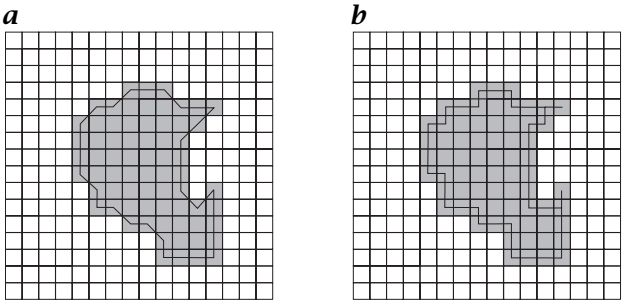

Figure 19.4: *Boundary representation with the chain code:* ***a*** *8-neighborhood;* ***b*** *4-neighborhood.*

in a clockwise direction. In a 4-neighborhood there are 4 and in an 8-neighborhood 8 possible directions to go, which we can encode with a 3-bit or 2-bit code as indicated in Fig. 19.3. The boundary and its code extracted in this way is shown in Fig. 19.4 for a 4-neighborhood and an 8-neighborhood.

The chain code shows a number of obvious advantages over the matrix representation of a binary object:

First, the chain code is a compact representation of a binary object. Let us assume a disk-like object with a diameter of R pixels. In a direct matrix representation we need to store the *bounding box* of the object (see Section 19.5.4), i. e., about R^2 pixels which are stored in R^2 bits. The bounding rectangle is the smallest rectangle enclosing the object. If we use an 8-connected boundary, the disk has about πR boundary points. The chain code of the πR points can be stored in about $3\pi R$ bits. For objects with a diameter larger than 10, the chain code is a more compact representation.

Second, the chain code is a *translation invariant* representation of a binary object. This property makes it easier to compare objects. However, the chain code is neither rotation nor scale invariant. This is a significant disadvantage for object recognition, although the chain code can still be used to extract rotation invariant parameters, such as the area of the object.

Third, the chain code is a complete representation of an object or curve. Therefore, we can — at least in principle — compute any shape feature from the chain code.

As shown in Section 19.5, we can compute a number of shape parameters — including the perimeter and area — more efficiently using the chain-code representation than in the matrix representation of the binary image. The limitation here is, of course, that the chain code is a digital curve on a discrete grid and as such describes the boundary of the object only within the precision of the discrete grid.

If the object is not connected or if it has holes, we need more than one chain code to represent it. We must also include information on whether the boundary surrounds an object or a hole. Reconstruction of the binary image from a chain code is an easy procedure: we can draw the outline of the object and then use a *fill operation* to paint it.

19.3 Moment-Based Shape Features

19.3.1 Definitions

In this section we present a systematic approach to object shape description. We first define *moments* for gray value and binary images and then show how to extract useful shape parameters from this approach. We will discuss Fourier descriptors in a similar manner in Section 19.4.

We used moments in Section 3.2.2 to describe the probability density function for gray values. Here we extend this description to two dimensions and define the moments of the gray value function $g(\boldsymbol{x})$ of an object as

$$\mu_{p,q} = \int (x_1 - \overline{x_1})^p (x_2 - \overline{x_2})^q \, g(\boldsymbol{x}) \mathrm{d}^2 x, \tag{19.1}$$

where

$$\overline{x_i} = \int x_i g(\boldsymbol{x}) \mathrm{d}^2 x \Big/ \int g(\boldsymbol{x}) \mathrm{d}^2 x. \tag{19.2}$$

The integration includes the area of the object. Instead of the gray value, we may use more generally any pixel-based feature to compute object moments. The vector $\overline{\boldsymbol{x}} = (\overline{x_1}, \overline{x_2})$ is called the *center of mass* of the object by analogy to classical mechanics. Think of $g(\boldsymbol{x})$ as the density $\rho(\boldsymbol{x})$ of the object; then the zero-order moment $\mu_{0,0}$ becomes the total mass of the object.

All the moments defined in Eq. (19.1) are related to the center of mass. Therefore they are often denoted as *central moments*. Central moments are translation invariant and thus are useful features for describing the shape of objects.

For discrete binary images, the moment calculation reduces to

$$\mu_{p,q} = \sum (x_1 - \overline{x_1})^p (x_2 - \overline{x_2})^q. \tag{19.3}$$

The summation includes all pixels belonging to the object. For the description of object shape we may use moments based on either binary, gray scale or feature images. Moments based on gray scale or feature images reflect not only the geometrical shape of an object but also the distribution of features within the object. As such, they are generally different from moments based on binary images.

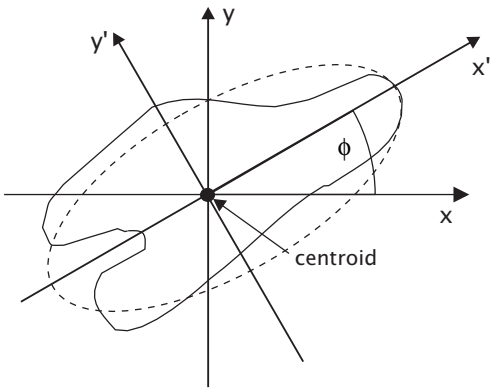

Figure 19.5: *Principal axes of the inertia tensor of an object for rotation around the center of mass.*

19.3.2 Scale-Invariant Moments

Often it is necessary to use shape parameters that do not depend on the size of the object. This is always required if objects observed from different distances must be compared. Moments can be normalized in the following way to obtain *scale-invariant* shape parameters. If we scale an object $g(\boldsymbol{x})$ by a factor of α, $g'(\boldsymbol{x}) = g(\boldsymbol{x}/\alpha)$, its moments are scaled by

$$\mu'_{p,q} = \alpha^{p+q+2}\,\mu_{p,q}.$$

We can then normalize the moments with the zero-order moment, $\mu_{0,0}$, to gain *scale-invariant moments*

$$\bar{\mu} = \frac{\mu_{p,q}}{\mu_{0,0}^{(p+q+2)/2}}.$$

Because the zero-order moment of a binary object gives the area of the object (Eq. (19.3)), the normalized moments are scaled by the area of the object. Second-order moments ($p + q = 2$), for example, are scaled with the square of the area.

19.3.3 Moment Tensor

Shape analysis beyond area measurements starts with the second-order moments. The zero-order moment just gives the area or "total mass" of a binary or gray value object, respectively. The first-order central moments are zero by definition.

The analogy to mechanics is again helpful to understand the meaning of the second-order moments $\mu_{2,0}$, $\mu_{0,2}$, and $\mu_{1,1}$. They contain terms in which the gray value function, i. e., the density of the object, is multiplied

by squared distances from the center of mass. Exactly the same terms are also included in the inertia tensor that was discussed in Section 13.3.7 (see Eqs. (13.23) and (13.24)). The three second-order moments form the components of the *inertia tensor* for rotation of the object around its center of mass:

$$J = \begin{bmatrix} \mu_{2,0} & -\mu_{1,1} \\ -\mu_{1,1} & \mu_{0,2} \end{bmatrix}. \tag{19.4}$$

Because of this analogy, we can transfer all the results from Section 13.3 to shape description with second-order moments. The *orientation* of the object is defined as the angle between the x axis and the axis around which the object can be rotated with minimum inertia. This is the eigenvector of the minimal eigenvalue. The object is most elongated in this direction (Fig. 19.5). According to Eq. (13.12), this angle is given by

$$\phi = \frac{1}{2} \arctan \frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}}. \tag{19.5}$$

As a measure for the *eccentricity* ε, we can use what we have defined as a coherence measure for local orientation Eq. (13.15):

$$\varepsilon = \frac{(\mu_{2,0} - \mu_{0,2})^2 + 4\mu_{1,1}^2}{(\mu_{2,0} + \mu_{0,2})^2}. \tag{19.6}$$

The eccentricity ranges from 0 to 1. It is zero for a circular object and one for a line-shaped object. Thus, it is a better-defined quantity than circularity with its odd range (Section 19.5.3).

Shape description by second-order moments in the *moment tensor* essentially models the object as an *ellipse*. The combination of the three second-order moments into a tensor nicely results in two rotation-invariant terms, the trace of the tensor, or $\mu_{2,0} + \mu_{0,2}$, which gives the radial distribution of features in the object, and the eccentricity Eq. (19.6), which measures the roundness, and one term which measures the orientation of the object. Moments allow for a complete shape description [148]. The shape description becomes more detailed the more higher-order moments are used.

19.4 Fourier Descriptors

19.4.1 Cartesian Fourier Descriptors

Fourier descriptors, like the chain code, use only the boundary of the object. In contrast to the chain code, Fourier descriptors do not describe curves on a discrete grid. They can be formulated for continuous or sampled curves. Consider the closed boundary curve sketched in Fig. 19.6. We can describe the boundary curve in a parametric description by taking the path length p from a starting point $[x_0, y_0]^T$ as a parameter.

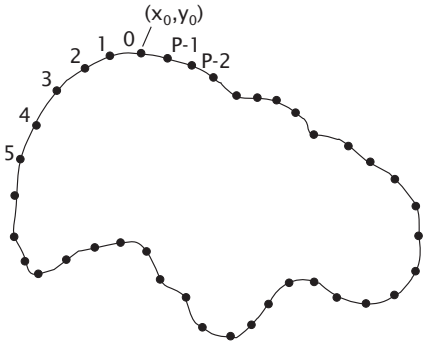

Figure 19.6: *Illustration of a parametric representation of a closed curve. The parameter p is the path length from the starting point* $[x_0, y_0]^T$ *in the counterclockwise direction. An equidistant sampling of the curve with P points is also shown.*

It is not easy to generate a boundary curve with equidistant samples. Discrete boundary curves, like the chain code, have significant disadvantages. In the 8-neighborhood, the samples are not equidistant. In the 4-neighborhood, the samples are equidistant, but the boundary is jagged because the pieces of the boundary curve can only go in horizontal or vertical directions. Therefore, the perimeter tends to be too long. Consequently, it does not seem a good idea to form a continuous boundary curve from points on a regular grid. The only alternative is to extract subpixel-accurate object boundary curves directly from the gray scale images. But this is not an easy task. Thus, the accurate determination of Fourier descriptors from contours in images still remains a challenging research problem.

The continuous boundary curve is of the form $x(p)$ and $y(p)$. We can combine these two curves into one curve with the complex function $z(p) = x(p) + \mathrm{i}y(p)$. This curve is cyclic. If P is the perimeter of the curve, then

$$z(p + nP) = z(p) \quad n \in \mathbb{Z}. \tag{19.7}$$

A cyclic or periodic curve can be expanded in a *Fourier series* (see also Table 2.1). The coefficients of the Fourier series are given by

$$\hat{z}_v = \frac{1}{P} \int_0^P z(p) \exp\left(\frac{-2\pi \mathrm{i} v p}{P}\right) \mathrm{d}p \quad v \in \mathbb{Z}. \tag{19.8}$$

The periodic curve can be reconstructed from the Fourier coefficients by

$$z(p) = \sum_{v=-\infty}^{\infty} \hat{z}_v \exp\left(\frac{2\pi \mathrm{i} v p}{P}\right). \tag{19.9}$$

The coefficients $\hat{z}_v$ are known as the *Cartesian Fourier descriptors* of the boundary curve. Their meaning is straightforward. The first coefficient

$$\hat{z}_0 = \frac{1}{P}\int_0^P z(p)\mathrm{d}p = \frac{1}{P}\int_0^P x(p)\mathrm{d}p + \frac{\mathrm{i}}{P}\int_0^P y(p)\mathrm{d}p \tag{19.10}$$

gives the mean vortex or *centroid* of the boundary. The second coefficient describes a circle

$$z_1(p) = \hat{z}_1 \exp\left(\frac{2\pi \mathrm{i} p}{P}\right) = r_1 \exp\left(\mathrm{i}\varphi_1 + 2\pi \mathrm{i} p/P\right). \tag{19.11}$$

The radius r_1 and the starting point at an angle φ_1 are given by $\hat{z}_1 = r_1 \exp(\mathrm{i}\varphi_1)$. The coefficient $\hat{z}_{-1}$ also results in a circle

$$z_{-1}(p) = r_{-1} \exp\left(\mathrm{i}\varphi_{-1} - 2\pi \mathrm{i} p/P\right), \tag{19.12}$$

but this circle is traced in the opposite direction (clockwise). With both complex coefficients together — in total four parameters — an ellipse can be formed with arbitrary half-axes a and b, orientation ϑ of the main axis a, and starting angle φ_0 on the ellipses. As an example, we take $\varphi_1 = \varphi_{-1} = 0$. Then,

$$z_1 + z_{-1} = (r_1 + r_{-1}) \cdot \cos\left(\frac{2\pi p}{P}\right) + \mathrm{i}(r_1 - r_{-1}) \sin\left(\frac{2\pi p}{P}\right). \tag{19.13}$$

This curve has the parametric form of an ellipse where the axes lie along the coordinate axes and the starting point is on the x axis.

From this discussion it is obvious that Fourier descriptors must always be paired. The pairing of higher-order coefficients also results in ellipses. These ellipses, however, are cycled n times. Added to the basic ellipse of the first pair, this means that the higher-order Fourier descriptors add more and more details to the boundary curve.

For further illustration, the reconstruction of the letters T and L is shown with an increasing number of Fourier descriptors (Fig. 19.7). The example show that only a few coefficients are required to describe even quite complex shapes.

Fourier descriptors can also be computed easily from sampled boundaries z_n. If the perimeter of the closed curve is P, N samples must be taken at equal distances of P/N (Fig. 19.6). Then,

$$\hat{z}_v = \frac{1}{N}\sum_{n=0}^{N-1} z_n \exp\left(-\frac{2\pi \mathrm{i} n v}{N}\right). \tag{19.14}$$

All other equations are valid also for sampled boundaries. The sampling has just changed the Fourier series to a discrete Fourier transform with only N wave number coefficients that run from 0 to $N-1$ or from $-N/2$ to $N/2-1$ (see also Table 2.1).

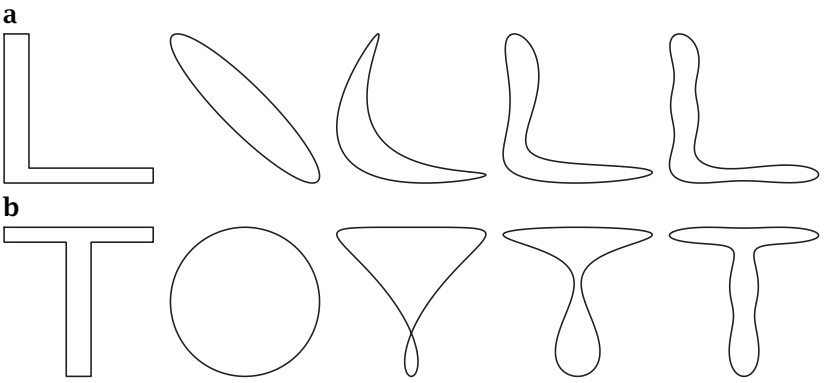

Figure 19.7: *Reconstruction of shape of* ***a*** *the letter "L" and* ***b*** *the letter "T" with 2, 3, 4, and 8 Fourier descriptor pairs.*

19.4.2 Polar Fourier Descriptors

An alternative approach to Fourier descriptors uses another parameterization of the boundary line. Instead of the path length p, the angle θ between the radius drawn from the centroid to a point on the boundary and the x axis is used. Thus, we directly describe the *radius* of the object as a function of the angle. Now we need only a real-valued sequence, $\boldsymbol{r}$, with N equiangular samples to describe the boundary. The coefficients of the discrete Fourier transform of this sequence,

$$\hat{r}_v = \frac{1}{N}\sum_{n=0}^{N-1} r_n \exp\left(-\frac{2\pi i n v}{N}\right), \tag{19.15}$$

are known as the *polar Fourier descriptors* of the boundary. Here, the first coefficient, $\hat{r}_0$, is equal to the mean radius. Polar Fourier descriptors cannot be used for all types of boundaries. The radial boundary parameterization $r(\theta)$ must be single-valued. Because of this significant restriction, we focus the further discussion of Fourier descriptors on Cartesian Fourier descriptors.

19.4.3 Symmetric Objects

Symmetries can easily be detected in Fourier descriptors. If a contour has *m-rotational symmetry*, then only $z_{1\pm vm}$ can be unequal to zero. This is demonstrated in Fig. 19.8 with the Fourier descriptors of a vertical line, a triangle, and a square. If one contour is the mirror contour of another, their Fourier descriptors are conjugate complex to each other.

The Fourier descriptors can also be used for *non-closed boundaries*. To make them closed, we simply trace the curve backward and forward.

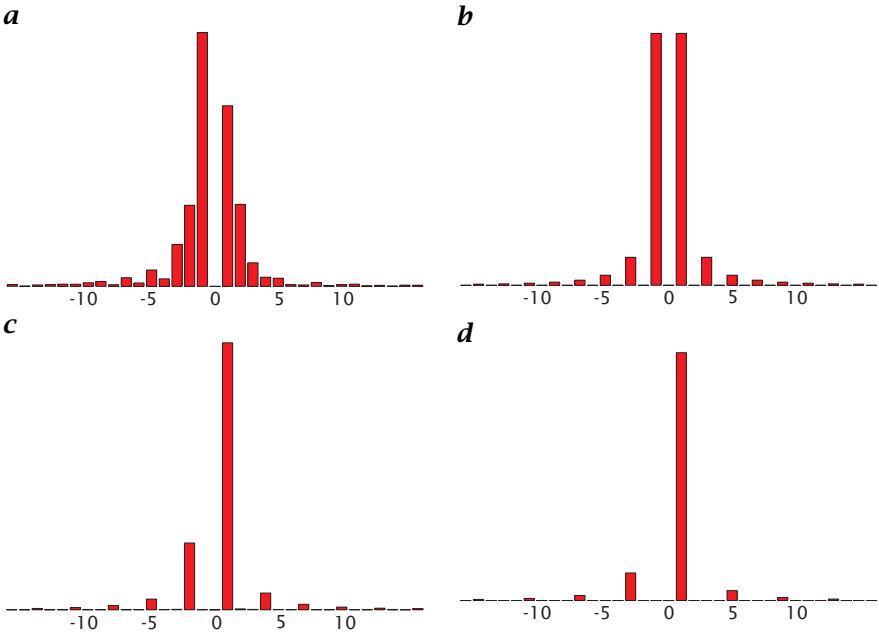

Figure 19.8: *Influence of the symmetry of an object on its Fourier descriptors:* ***a*** *the letter L,* ***b*** *a line,* ***c*** *a triangle, and* ***d*** *a square. Shown are the magnitude of the Fourier descriptors from* $v = -16$ *to* $v = 16$.

It is easy to recognize such curves, as their area is zero. From Eq. (19.17), we can then conclude that $|\hat{z}_{-v}| = |\hat{z}_v|$. If the trace begins at one of the endpoints, even $\hat{z}_{-v} = \hat{z}_v$.

19.4.4 Invariant Object Description

Translation invariance. The position of the object is confined to a single coefficient $\hat{z}_0$. All other coefficients are translation invariant.

Scale invariance. If the contour is scaled by a coefficient α, all Fourier descriptors are also scaled by α. For an object with non-zero area, and if the contour is traced counterclockwise, the first coefficient is always unequal to zero. Thus, we can simply scale all Fourier descriptors by $|\hat{z}_1|$ to obtain scale invariant shape descriptors. Note that these scaled descriptors are still complete.

Rotation invariance. If a contour is rotated counter clockwise by the angle φ_0, the Fourier descriptor $\hat{z}_v$ is multiplied by the phase factor $\exp(\mathrm{i}v\varphi_0)$ according to the *shift theorem* for the Fourier transform (Theorem 3, p. 52, ≻R4). The shift theorem makes the construction of

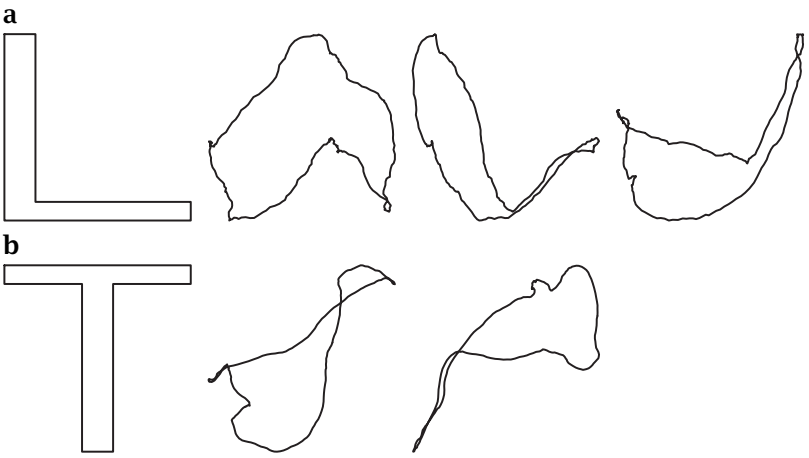

Figure 19.9: *Importance of the phase for the description of shape with Fourier descriptors. Besides the original letters, three random modifications of the phase are shown with unchanged magnitude of the Fourier descriptors.*

rotation-invariant Fourier descriptors easy. For example, we can relate the phases of all Fourier descriptors to the phase of $\hat{z}_1$, φ_1, and subtract the phase shift $v\varphi_1$ from all coefficients. Then, all remaining Fourier descriptors are rotation invariant.

Both Fourier descriptors (Section 19.4) and moments (Section 19.3) provide a framework for scale and rotation invariant shape parameters. The Fourier descriptors are the more versatile instrument. However, they restrict the object description to the boundary line while moments of gray scale objects are sensitive to the spatial distribution of the gray values in the object.

Ideally, form parameters describe the form of an object completely and uniquely. This means that different shapes must not be mapped onto the same set of features. A scale and rotation invariant but incomplete shape description is given by the magnitude of the Fourier descriptors. Figure 19.9 shows how different shapes are mapped onto this shape descriptor by taking the Fourier descriptors of the letters "T" and "L" and changing the phase randomly.

Only the complete set of the Fourier descriptors constitutes a unique shape description. Note that for each invariance, one degree of freedom is lost. For translation invariance, we leave out the first Fourier descriptor $\hat{z}_0$ (two degrees of freedom). For scale invariance, we set the magnitude of the second Fourier descriptor, $\hat{z}_1$, to one (one degree of freedom), and for rotation invariance, we relate all phases to the phase

of $\hat{z}_1$ (another degree of freedom). With all three invariants, four degrees of freedom are lost.

It is the beauty of the Fourier descriptors that these invariants are simply contained in the first two Fourier descriptors. If we norm all other Fourier descriptors with the phase and magnitude of the second Fourier descriptor, we have a complete translation, rotation, and scale invariant description of the shape of objects. By leaving out higher-order Fourier descriptors, we can gradually relax fine details from the shape description in a controlled way.

Shape differences can then be measured by using the fact that the Fourier descriptors form a complex-valued vector. A metric for shape differences is then given by the magnitude of the difference vector:

$$d_{zz'} = \sum_{v=-N/2}^{N/2-1} |\hat{z}_v - \hat{z}'_v|^2 . \tag{19.16}$$

Depending on which normalization we apply to the Fourier descriptors, this metric will be scale and/or rotation invariant.

19.5 Shape Parameters

After discussing different ways to represent binary objects extracted from image data, we now turn to the question of how to describe the shape of these objects. This section discusses elementary geometrical parameters such as area and perimeter.

19.5.1 Area

One of the most trivial shape parameters is the *area A* of an object. In a digital binary image the area is given by the number of pixels belonging to the image. So in the matrix or pixel list representation of the object, computing its area simply means counting the number of pixels. The area is also given as the zero-order moment of a binary object (Eq. (19.3)).

At first glance, area computation of an object which is described by its chain-code seems to be a complex operation. However, the contrary is true. Computation of the area from the chain code is much faster than counting pixels as the boundary of the object contains only a small fraction of the object's pixels and requires only two additions per boundary pixel.

The algorithm works in a similar way as numerical integration. We assume a horizontal base line drawn at an arbitrary vertical position in the image. Then we start the integration of the area at the uppermost pixel of the object. The distance of this point to the base line is B. We follow the boundary of the object and increment the area of the object according to the figures in Table 19.1.

Table 19.1: *Computation of the area of an object from the chain code. Initially, the area is set to 1. With each step, the area and the parameter B are incremented corresponding to the value of the chain code; after Zamperoni [202].*

Contour code	Area increment	Increment of B
0	$+B$	0
1	$+B$	1
2	0	1
3	$-B+1$	1
4	$-B+1$	0
5	$-B+1$	-1
6	0	-1
7	$+B$	-1

If we, for example, move to the right (chain code 0), the area increases by B. If we move upwards to the right (chain code 1), the area also increases by B, but B must also be incremented, because the distance between the boundary pixel and the base line has increased. For all movements to the left, the area is decreased by $B-1$. The value $B-1$ has to be used instead of B because an object part with the height of one pixel adds an area of one and not of zero. In this way, we subtract the area between the lower boundary line of the object and the base line, which was included in the area computation when moving to the right. The area A is initially set to 1.

The area can also be computed from Fourier descriptors. It is given by

$$A = \pi \sum_{v=-N/2}^{N/2-1} v\,|\hat{z}_v|^2. \tag{19.17}$$

This is a fast algorithm, which requires at most as many operations as points on the boundary line of the curve. The Fourier descriptors have the additional advantage that we can compute the area for a certain degree of smoothness by taking only a certain number of Fourier descriptors. The more Fourier descriptors we take, the more detailed is the boundary curve, as demonstrated in Fig. 19.7.

19.5.2 Perimeter

The *perimeter* is another geometrical parameter that can easily be obtained from the chain code of the object boundary. We just need to count the length of the chain code and take into consideration that steps in diagonal directions are longer by a factor of $\sqrt{2}$. The perimeter p is then

given by an 8-neighborhood chain code:

$$p = n_e + \sqrt{2} n_o, \tag{19.18}$$

where n_e and n_o are the number of even and odd chain code steps, respectively. The steps with an uneven code are in a diagonal direction.

In contrast to the area, the perimeter is a parameter that is sensitive to the noise level in the image. The noisier the image, the more rugged and thus longer the boundary of an object will become in the segmentation procedure. This means that care must be taken in comparing perimeters that have been extracted from different images. We must be sure that the smoothness of the boundaries in the images is comparable.

Unfortunately, no simple formula exists to compute the perimeter from the Fourier descriptors, because the computation of the perimeter of ellipses involves elliptic integrals. However, the perimeter results directly from the construction of the boundary line with equidistant samples and is well approximated by the number of sampling points times the mean distance between the points.

19.5.3 Circularity

Area and perimeter are two parameters which describe the size of an object in one or the other way. In order to compare objects observed from different distances, it is important to use shape parameters that do not depend on the size of the object on the image plane. The *circularity* c is one of the simplest parameters of this kind. It is defined as

$$c = \frac{p^2}{A}. \tag{19.19}$$

The circularity is a dimensionless number with a minimum value of $4\pi \approx 12.57$ for circles. The circularity is 16 for a square and $12\sqrt{3} \approx 20.8$ for an equilateral triangle. Generally, it tends towards large values for elongated objects.

Area, perimeter, and circularity are shape parameters which do not depend on the orientation of the objects on the image plane. Thus they are useful to distinguish objects independently of their orientation.

19.5.4 Bounding Box

Another simple and useful parameter for a crude description of the size of an object is the bounding box. It is defined as the rectangle that is just large enough to contain all object pixels. It gives also a rough description of the shape of the object. In contrast to the area (Section 19.5.1), however, it is not rotation invariant. It can be made rotation invariant if the object is first rotated into a standard orientation, for instance using the orientation of the moment tensor (Section 19.3.3). In any case,

the bounding box is a useful feature if any further object-oriented pixel processing is required, such as extraction of the object pixels for further reference purposes.

19.6 Further Readings‡

Spatial data structures, especially various tree structures, and their applications are detailed in the monographs by Samet [159, 160]. A detailed discussion of moment-based shape analysis with emphasize on invariant shape features can be found in the monography of Reiss [148]. Invariants based on gray values are discussed by Burkhardt and Siggelkow [14].

20 Classification

20.1 Introduction

When objects are detected with suitable operators and their shape is described (Chapter 19), image processing has reached its goal for certain classes of applications. For other applications, further tasks remain to be solved. In this introduction we explore several examples which illustrate how the image processing tasks depend on the questions we pose.

In many image processing applications, the size and shape of particles such as bubbles, aerosols, drops, pigment particles, or cell nuclei must be analyzed. In these cases, the parameters of interest are clearly defined and directly measurable from the images taken. We determine the area and shape of each particle detected with the methods discussed in Sections 19.5.1 and 19.3. Knowing these parameters allows all the questions of interest to be answered. From the data collected, we can, for example, compute histograms of the particle area (Fig. 20.1c). This example is typical for a wide class of scientific applications. Object parameters that can be evaluated directly and unambiguously from the image data help to answer the scientific questions asked.

Other applications are more complex in the sense that it is required to distinguish different classes of objects in an image. The easiest case is given by a typical industrial inspection task. Are the dimensions of a part within the given tolerance? Are any parts missing? Are any defects such as scratches visible? As the result of the analysis, the inspected part either passes the test or is assigned to a certain error class.

Assigning objects in images to certain classes is — like many other aspects of image processing and analysis — a truly interdisciplinary problem which is not specific to image analysis but a very general type of technique. In this respect, image analysis is part of a more general research area known as *pattern recognition*. A classical application of pattern recognition that everybody knows is *speech recognition*. The spoken words are contained in a 1-D acoustic signal (a time series). Here, the classification task is to recognize the phonemes, words, and sentences from the spoken language. The corresponding task in image processing is *text recognition*, the recognition of letters and words from a written text, also known as *optical character recognition* (*OCR*).

A general difficulty of classification is related to the fact that the relationship between the parameters of interest and the image data is

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2

Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

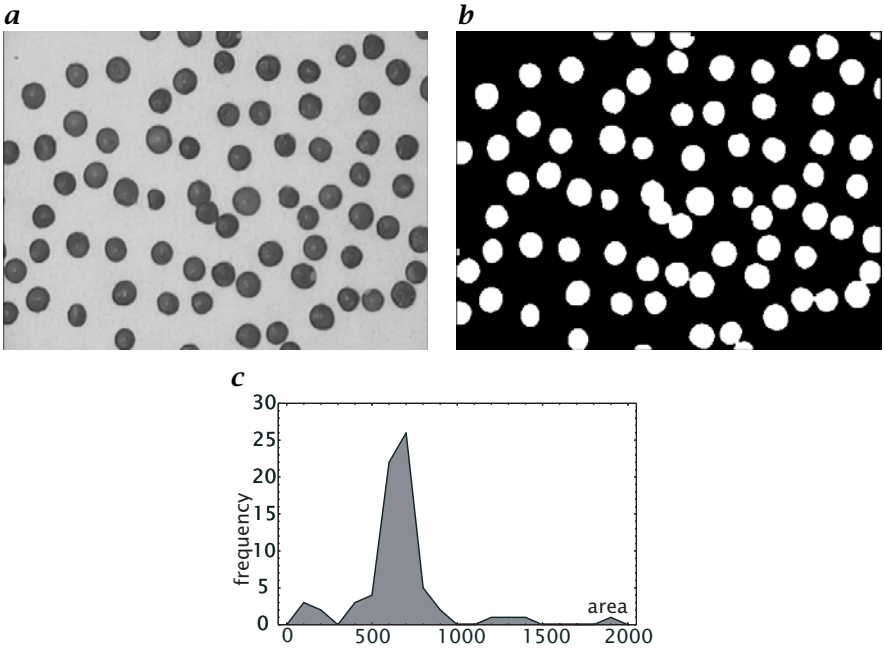

Figure 20.1: *Steps to analyze the size distribution of particles (lentils):* ***a*** *original image,* ***b*** *binary image, and* ***c*** *area distribution.*

not evident. The objects to be classified are not directly related to a certain range of values of a single feature but have to be identified by their optical signature in the image. By which features, for example, can we distinguish the lentils, peppercorns, and sunflower seeds shown in Fig. 20.2? The relation between the optical signatures and the object classes generally requires a careful investigation. We illustrate the complex relations between object features and their optical signatures with two further examples.

"*Waldsterben*" (large-scale forest damage by acid rain and other environmental pollution) is one of the many large problems environmental scientists are faced with. In remote sensing, the task is to map and classify the extent of the damage in forests from *aerial* and *satellite imagery*. In this example, the relationship between the different classes of damage and features in the images is less evident. Detailed investigations are necessary to reveal these complex relationships. Aerial images must be compared with investigations on the ground. We can expect to need more than one feature to identify certain classes of forest damage.

There are many similar applications in medical and biological science. One of the standard questions in medicine is to distinguish between "healthy" and "diseased". Again, it is obvious that we cannot expect a

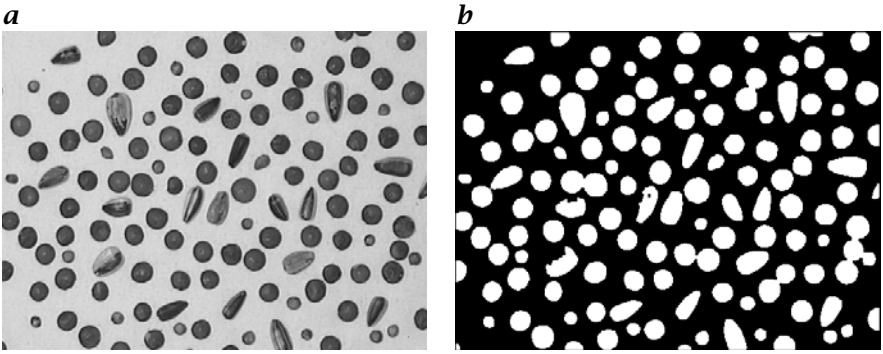

Figure 20.2: *Classification task: which of the seeds is a peppercorn, a lentil, a sunflower seed or none of the three?* ***a*** *Original image, and* ***b*** *binary image after segmentation.*

simple relationship between these two object classes and features of the observed objects in the images.

Take as another example the objects shown in Fig. 20.3. We will have no problem in recognizing that all objects but one are lamps. How could a machine vision system perform this task? What features can we extract from these images that help us recognize a lamp? While we have no problems in recognizing the lamps in Fig. 20.3, we feel quite helpless with the question as how we can solve this task using a computer. Obviously this task is complex. We recognize a lamp because we have already seen many other lamps before and somehow memorized this experience and are able to compare this stored knowledge with what we see in the image. But how is this knowledge stored and how is the comparison performed? It is obviously not just a database with geometric shapes, we also know in which context or environment lamps occur and for what they are used. Research on problems of this kind is part of a research area called *artificial intelligence*, abbreviated as *AI*.

With respect to scientific applications, another aspect of classification is of interest. As imaging techniques are among the driving forces of progress in experimental natural sciences, it often happens that unknown objects appear in images, for which no classification scheme is available so far. It is one goal of image processing to find out possible classes for these new objects. Therefore, we need classification techniques that do not require any previous knowledge.

Summing up, we conclude that classification includes two basic tasks:

1. The relation between the image features (*optical signature*) and the object classes sought must be investigated in as much detail as possible. This topic is partly comprised in the corresponding scientific

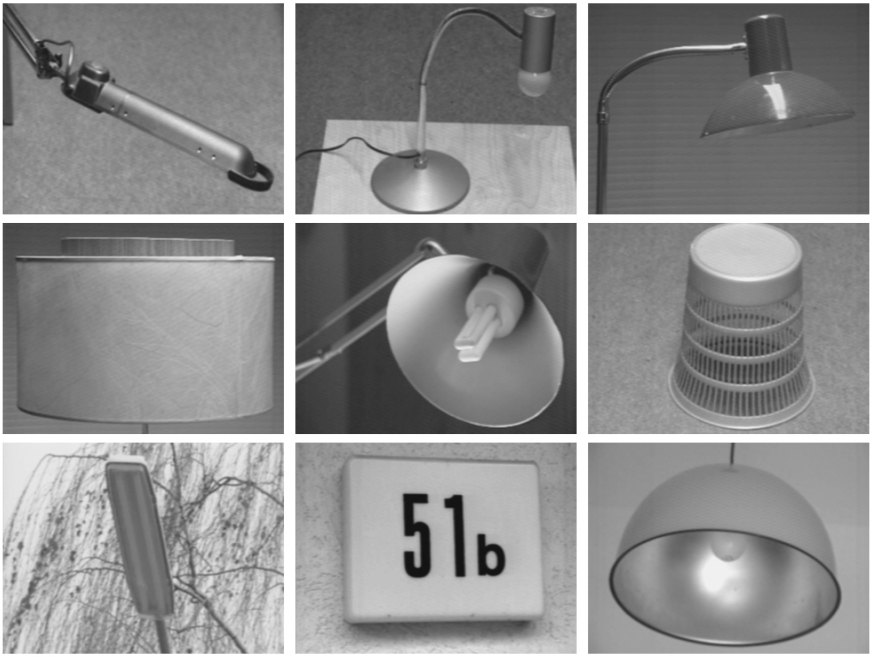

Figure 20.3: *How do we recognize that all but one of these objects are lamps?*

area and partly in image formation, i. e., optics, as discussed in Chapters 6–8.

2. From the multitude of possible image features, we must select an optimal set which allows the different object classes to be distinguished unambiguously with minimum effort and as few errors as possible by a suitable classification technique. This task, known as *classification*, is the topic of this chapter. We touch here only some basic questions such as selecting the proper type and number of features (Section 20.2) and devise some simple classification techniques (Section 20.3).

20.2 Feature Space

20.2.1 Pixel-Based Versus Object-Based Classification

Two types of classification procedures can be distinguished: *pixel-based classification* and *object-based classification*. In complex cases, a segmentation of objects is not possible using a single feature. Then it is already required to use multiple features and a classification process to decide which pixel belongs to which type of object.

A much simpler object-based classification can be used if the different objects can be well separated from the background and do not touch or overlap each other. Object-based classification should be used if at all possible, since much less data must be handled. Then all the pixel-based features discussed in Chapters 11–15, such as the mean gray value, local orientation, local wave number, and gray value variance, can be averaged over the whole area of the object and used as features describing the object's properties. In addition, we can use all the parameters describing the shape of the objects discussed in Chapter 19. Sometimes it is required to apply both classification processes: first, a pixel-based classification to separate the objects from each other and the background and, second, an object-based classification to utilize also the geometric properties of the objects for classification.

20.2.2 Cluster

A set of P features forms a P-dimensional space $\mathbb{M}$, denoted as a *feature space* or *measurement space*. Each pixel or object is represented as a *feature vector* in this space. If the features represent an object class well, all feature vectors of the objects from this class should lie close to each other in the feature space. We regard classification as a statistical process and assign a P-dimensional probability density function to each object class. In this sense, we can estimate this probability function by taking samples from a given object class, computing the feature vector, and incrementing the corresponding point in the discrete feature space. This procedure is that of a generalized P-dimensional *histogram* (Section 3.2.1). When an object class shows a narrow probability distribution in the feature space, we speak of a *cluster*. It will be possible to separate the objects into given object classes if the clusters for the different object classes are well separated from each other. With less suitable features, the clusters overlap each other or, even worse, no clusters may exist at all. In these cases, an error-free classification is not possible.

20.2.3 Feature Selection

We start with an example, the classification of the different seeds shown in Fig. 20.2 into the three classes peppercorns, lentils, and sunflower seeds. Figure 20.4a, b shows the histograms of the two features area and eccentricity (Eq. (19.6) in Section 19.3.3). While the area histogram shows two peaks, only one peak can be observed in the histogram of the eccentricity. In any case, neither of the two features alone is sufficient to distinguish the three classes peppercorns, lentils, and sunflower seeds. If we take both parameters together, we can identify at least two clusters (Fig. 20.4c). These two classes can be identified as the peppercorns and the lentils. Both seeds are almost circular and thus show a low eccen-

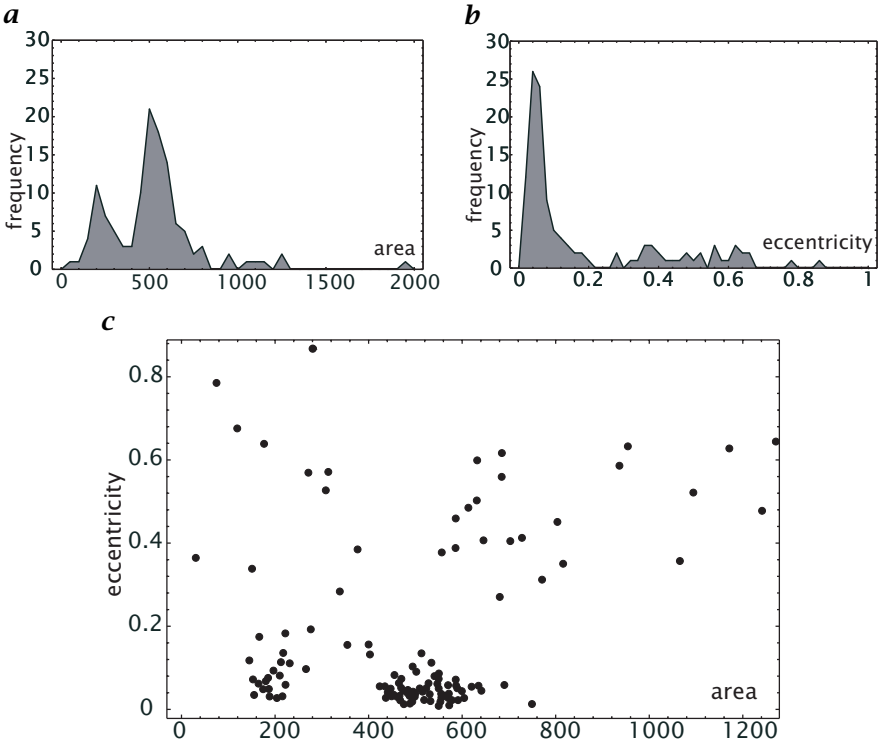

Figure 20.4: *Features for the classification of different seeds from Fig. 20.2 into peppercorns, lentils, and sunflower seeds: histogram of the features* ***a*** *area and* ***b*** *eccentricity;* ***c*** *two-dimensional feature space with both features.*

tricity between 0 and 0.2. Therefore, both classes coalesce into one peak in the eccentricity histogram (Fig. 20.4b). The sunflower seeds do not form a dense cluster since they vary significantly in shape and size. But it is obvious that they can be similar in size to the lentils. Thus it is not sufficient to use only the feature area.

In Figure 20.4c we can also identify several outliers. First, there are several small objects with high eccentricity. These are objects that are only partly visible at the edges of the image (Fig. 20.2). There are also five large objects where touching lentils merge into larger objects. The eccentricity of these objects is also large and it may be impossible to distinguish them from sunflower seeds using the two simple parameters area and eccentricity only.

The quality of the features is critical for a good classification. What does this mean? At first glance, we might think that as many features as possible would be the best solution. Generally, this is not the case. Figure 20.5a shows a one-dimensional feature space with three object

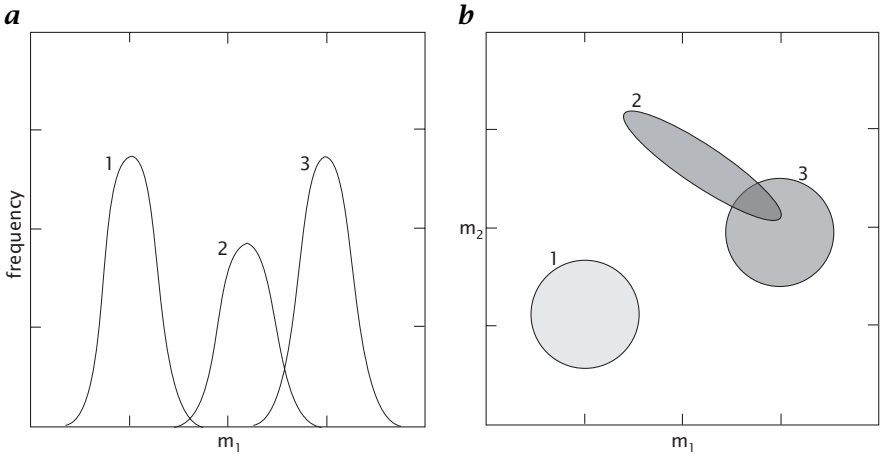

Figure 20.5: ***a*** *One-dimensional feature space with three object classes.* ***b*** *Extension of the feature space with a second feature. The gray shaded areas indicate the regions in which the probability for a certain class is larger than zero. The same object classes are shown in* ***a*** *and* ***b****.*

classes. The features of the first and second class are separated, while those of the second and third class overlap considerably. A second feature does not necessarily improve the classification, as demonstrated in Fig. 20.5b. The clusters of the second and third class are still overlaid. A closer examination of the distribution in the feature space explains why: the second feature does not tell us much new. It varies in strong correlation with the first feature. Thus, the two features are strongly correlated.

Two additional basic facts are worth mentioning. It is often overlooked how many different classes can be separated with a few parameters. Let us assume that one feature can only separate two classes. Then, ten features can separate $2^{10} = 1024$ object classes. This simple example illustrates the high separation potential of just a few parameters. The essential problem is the even distribution of the clusters in the feature space. Consequently, it is important to find the right features, i. e., to study the relationship between the features of the objects and those in the images carefully.

20.2.4 Distinction of Classes in the Feature Space

Even if we take the best features available, there may be classes that cannot be separated. In such a case, it is always worth reminding us that separating the objects in well-defined classes is only a model of reality. Often, the transition from one class to another may not be abrupt but rather gradual. For example, anomalies in a cell may be present to a vary-

1990, Oil, Island, L7R 4A6

Figure 20.6: *Illustration of the recognition of letters with very similar shape such as the large 'O' and the figure '0', or the letters 'I' and 'l' and the figure '1'.*

ing degree, there not being two distinct classes, "normal" and "pathological", but rather a continuous transition between the two. Thus, we cannot expect to find well-separated classes in the feature space in every case. We can draw two conclusions. First, it is not guaranteed that we will find well-separated classes in the feature space, even if optimal features have been selected. Second, this situation may force us to reconsider the object classification. Two object classes may either in reality be only one class or the visualization techniques to separate them may be inadequate.

In another important application, *optical character recognition*, or *OCR*, we do have distinct classes. Each character is a well-defined class. While it is easy to distinguish most letters, some, e. g., the large 'O' and the figure '0', or the letters 'I' and 'l' and the figure '1', are very similar, i. e., lie close to each other in the feature space (Fig. 20.6). Such well-defined classes that hardly differ in their features, pose serious problems for the classification task.

How can we then distinguish the large letter 'O' from the figure '0' or an 'l' from a capital 'I'? We can give two answers to this question. First, the fonts can be redesigned to make letters better distinguishable from each other. Indeed, special font sets have been designed for automated character recognition.

Second, additional information can be brought into the classification process. This requires, however, that the classification does not stop at the level of individual letters; it must be advanced to the word level. Then, it is easy to establish rules for better recognition. One simple rule which helps to distinguish the letter 'O' from the figure '0' is that letters and figures are not mixed in a word. As a counterexample to this rule, take British or Canadian zip codes which contain a blend of letters and figures. Anybody who is not trained to read this unusual mix has serious problems in reading and memorizing them. As another example, the capital 'I' can be distinguished from the lowercase 'l' by the rule that capital letters occur only as the first letter in a word or in an all-capital-letter word.

We close this section with the comment that asking whether a classification is at all possible for a given problem either by its nature or by the type of possible features is at least as important, if not more so, than the proper selection of a classification method.

20.2.5 Principal Axes Transform

The discussion in the previous section suggested that we must choose the object features very carefully. Each feature should bring in new information which is orthogonal to what we already know about the object classes; i. e., object classes with a similar distribution in one feature should differ in another feature. In other words, the features should be uncorrelated. The correlation of features can be studied with the statistical methods discussed in Section 3.3, provided that the distribution of the features for the different classes is known (supervised classification).

The important quantity is the *cross-covariance* of two features m_p and m_q from the P-dimensional feature vector for *one* object class, which is defined as

$$C_{pq} = \overline{\left(m_p - \overline{m_p}\right)\left(m_q - \overline{m_q}\right)}. \tag{20.1}$$

If the cross-covariance C_{pq} is zero, the features are said to be uncorrelated or orthogonal. The variance

$$C_{pp} = \overline{\left(m_p - \overline{m_p}\right)^2} \tag{20.2}$$

is a measure for the variance of the feature. A good feature for a certain object class should show a small variance indicating a narrow extension of the cluster in the corresponding direction of the feature space. With P features, we can form a symmetric matrix with the coefficients C_{pq}, the *covariance matrix*

$$\boldsymbol{C} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1,P} \\ C_{12} & C_{22} & \dots & C_{2,P} \\ \vdots & \vdots & \ddots & \vdots \\ C_{1,P} & C_{2,P} & \dots & C_{P,P} \end{bmatrix}. \tag{20.3}$$

The diagonal elements of the covariance matrix contain the variances of the P features, while the off-diagonal elements constitute the cross-covariances. Like every symmetric matrix, the covariance matrix can be diagonalized (compare the discussion on the tensor representation of neighborhoods in Section 13.3). This procedure is called the *principal-axes transform*. The covariance matrix in the principal-axes coordinate system reads

$$\boldsymbol{C}' = \begin{bmatrix} C'_{11} & 0 & \cdots & 0 \\ 0 & C'_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & C'_{pp} \end{bmatrix}. \tag{20.4}$$

The diagonalization shows that we can find a new coordinate system in which all features are uncorrelated. Those new features are linear

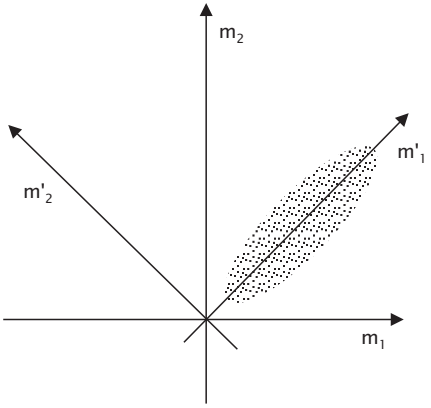

Figure 20.7: *Illustration of correlated features and the principal-axes transform.*

combinations of the old features and are the eigenvectors of the covariance matrix. The corresponding eigenvalues are the variances of the transformed features. The best features show the lowest variance; features with large variances are not of much help since they are spread out in the feature space and, thus, do not contribute much to separating different object classes. Thus, they can be omitted without making the classification significantly worse.

A trivial but illustrative example is the case when two features are nearly identical, as illustrated in Fig. 20.7. In this example, the features m_1 and m_2 for an object class are almost identical, since all points in the feature space are close to the main diagonal and both features show a large variance. In the principal-axes coordinate system $m_2' = m_1 - m_2$ is a good feature, as it shows a narrow distribution, while m_1' is as useless as m_1 and m_2 alone. Thus we can reduce the feature space from two dimensions to one without any disadvantages.

In this way, we can use the principal-axes transform to reduce the dimension of the feature space and find a smaller set of features which does nearly as good a job. This requires an analysis of the covariance matrix for all object classes. Only those features can be omitted where the analysis for *all* classes gives the same results. To avoid misunderstandings, the principal-axes transform cannot improve the separation quality. If a set of features cannot separate two classes, the same feature set transformed to the principal-axes coordinate system will not do so either. Given a set of features, we can only find an optimal subset and, thus, reduce the computational costs of classification.

20.2.6 Supervised and Unsupervised Classification

We can regard the classification problem as an analysis of the structure of the feature space. One object is thought of as a *pattern* in the feature space. Generally, we can distinguish between *supervised classification* and *unsupervised classification* procedures. Supervision of a classification procedure means determining the clusters in the feature space with known objects beforehand using teaching areas for identifying the clusters. Then, we know the number of classes and their location and extension in the feature space.

With unsupervised classification, no knowledge is presumed about the objects to be classified. We compute the patterns in the feature space from the objects we want to classify and then perform an analysis of the clusters in the feature space. In this case, we do not even know the number of classes beforehand. They result from the number of well-separated clusters in the feature space. Obviously, this method is more objective, but it may result in a less favorable separation.

Finally, we speak of *learning* methods if the feature space is updated by each new object that is classified. Learning methods can compensate any temporal trends in the object features. Such trends may be due to simple reasons such as changes in the illumination, which could easily occur in an industrial environment because of changes in daylight, aging, or dirtying of the illumination system.

20.3 Simple Classification Techniques

In this section, we will discuss different classification techniques. They can be used for both unsupervised and supervised classification. The techniques differ only by the method used to associate classes with clusters in the feature space (Section 20.2.6).

Once the clusters are identified by either method, the further classification process is identical for both of them. A new object delivers a feature vector that is associated with one of the classes or rejected as an unknown class. The different classification techniques differ only by the manner in which the clusters are modeled in the feature space.

Common to all *classifiers* is a many to one mapping from the feature space $\mathbb{M}$ to the *decision space* $\mathbb{D}$. The decision space contains Q elements, each corresponding to a class including a possible rejection class for unidentifiable objects. In the case of a deterministic decision, the elements in the decision space are binary numbers. Then only one of the elemets can be one; all others must be zero. If the classifiers generates a probabilistic decision, the elements in the decision space are real numbers. Then the sum of all elements in the decision space must be one.

20.3.1 Look-up Classification

This is the simplest classification technique but in some cases also the best, since it does not perform any modeling of the clusters for the different object classes, which can never be perfect. The basic approach of look-up classification is very simple. Take the feature space as it is and mark in every cell to which class it belongs. Normally, a significant amount of cells do not belong to any class and thus are marked with 0.

In case the clusters from two classes overlap, we have two choices. First, we can take that class which shows the higher probability at this cell. Second, we could argue that an error-free classification is not possible with this feature vector and mark the cell with zero. After this initialization of the feature space, the classification reduces to a simple look-up operation (Section 10.2.2). A feature vector m is taken and is looked up in the multidimensional look-up table to see which class, if any, it belongs to.

Without doubt, this is a fast classification technique which requires a minimum number of computations. The downside of the method — as with many other fast techniques — is that it requires huge amounts of memory for the look-up tables. An example: a three-dimensional feature space with only 64 bins per feature requires $64 \times 64 \times 64 = 1/4$ MB of memory — if no more than 255 classes are required so that one byte is sufficient to hold all class indices. We can conclude that the look-up table technique is only feasible for low-dimensional feature spaces. This suggests that it is worthwhile to reduce the number of features. Alternatively, features with a narrow distribution of feature values for all classes are useful, since then a rather small range of values and, thus, a small number of bins per feature sufficiently reduce the memory requirements.

20.3.2 Box Classification

The box classifier provides a simple modeling of the clusters in the feature space. A cluster of one class is modeled by a bounding box tightly surrounding the area covered by the cluster (Fig. 20.8). It is obvious that the box method is a rather crude modeling. If we assume that the clusters are multidimensional normal distributions, then the clusters have an elliptic shape. These ellipses fit rather well into the boxes when the axes of the ellipse are parallel to the axes of the feature space. In a two-dimensional feature space, for example, an ellipse with half-axes a and b has an area of πab, the surrounding box an area of $4ab$. This is not too bad.

When the features are correlated with each other the clusters become long and narrow objects along diagonals in the feature space. Then the boxes contain a lot of void space and they tend much more easily to overlap, making classification impossible in the overlapping areas. How-

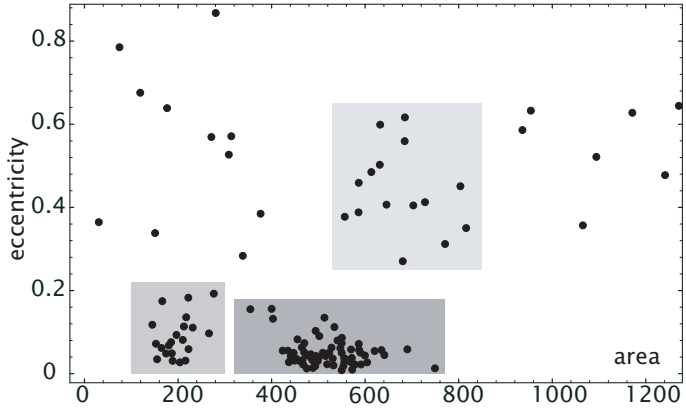

Figure 20.8: *Illustration of the box classifier for the classification of different seeds from Fig. 20.2 into peppercorns, lentils, and sunflower seeds using the two features area and eccentricity.*

Table 20.1: *Parameters and results of the simple box classification for the seeds shown in Fig. 20.2. The corresponding feature space is shown in Fig. 20.8.*

	Area	Eccentricity	Number
total	—	—	122
peppercorns	100–300	0.0–0.22	21
lentils	320–770	0.0–0.18	67
sunflower seeds	530–850	0.25–0.65	15
rejected			19

ever, correlated features can be avoided by applying of the principal-axes transform (Section 20.2.5).

The computations required for the box classifier are still modest. For each class and for each dimension of the feature space, two comparison operations must be performed to decide whether a feature vector belongs to a class or not. Thus, the maximum number of comparison operations for Q classes and a P-dimensional feature space is $2PQ$. In contrast, the look-up classifier required only P address calculations; the number of operations did not depend on the number of classes.

To conclude this section we discuss a realistic classification problem. Figure 20.2 showed an image with three different seeds, namely sunflower seeds, lentils, and peppercorns. This simple example shows many properties which are typical for a classification problem. Although the three classes are well defined, a careful consideration of the features to be used for classification is necessary since it is not immediately evi-

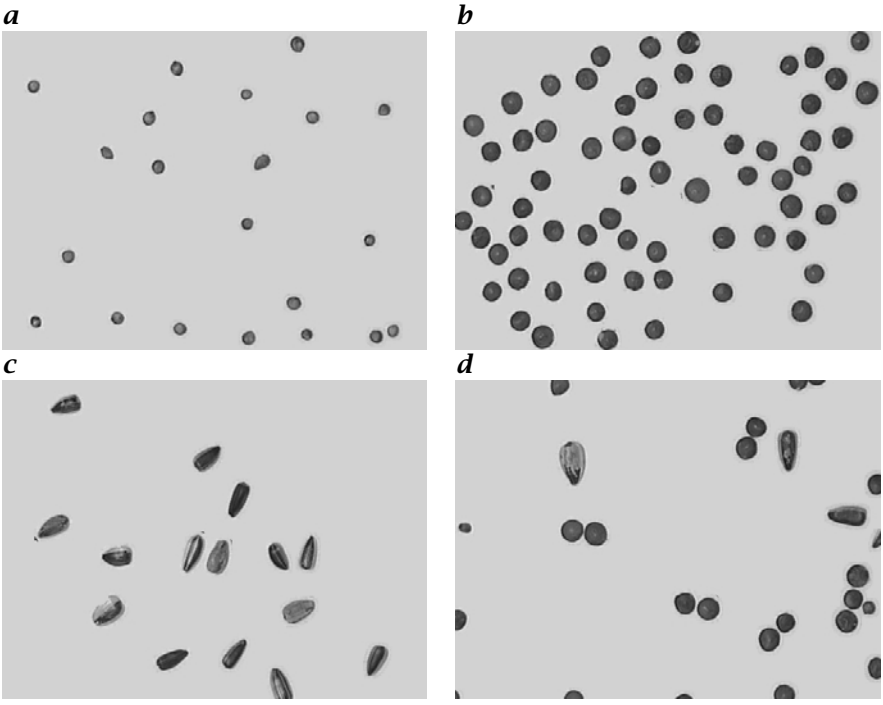

Figure 20.9: *Masked classified objects from image Fig. 20.2 showing the classified* ***a*** *peppercorns,* ***b*** *lentils,* ***c*** *sunflower seeds, and* ***d*** *rejected objects.*

dent which parameters can be successfully used to distinguish between the three classes. Furthermore, the shape of the seeds, especially the sunflower seeds, shows considerable fluctuations. The feature selection for this example was already discussed in Section 20.2.3.

Figure 20.8 illustrates the box classification using the two features area and eccentricity. The shaded rectangles mark the boxes used for the different classes. The conditions for the three boxes are summarized in Table 20.1. As the final result of the classification, Fig. 20.9 shows four images. In each of the images, only objects belonging to one of the subtotals from Table 20.1 are masked out. From a total of 122 objects, 103 objects were recognized. Thus 19 objects were rejected. They could not be assigned to any of the three classes for one of the following reasons:

- Two or more objects were so close to each other that they merged into one object. Then the values of the area and/or the eccentricity are too high.

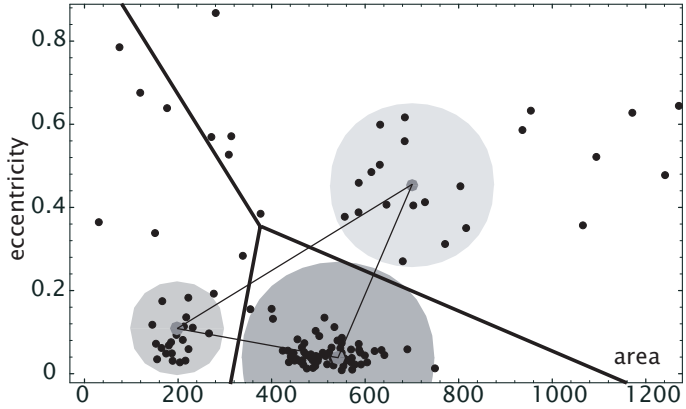

Figure 20.10: *Illustration of the minimum distance classifier with the classification of different seeds from Fig. 20.2 into peppercorns, lentils, and sunflower seeds using the two features area and eccentricity. A feature vector belongs to the cluster to which it has the minimal distance to the cluster center.*

- The object was located at the edge of the image and thus was only partly visible. This leads to objects with relatively small area but high eccentricity.
- Three large sunflower seeds were rejected because of too large an area. If we increased the area for the sunflower seed class then also merged lentils would be recognized as sunflower seeds. Thus this classification error can only be avoided if we avoid the merging of objects with a more advanced segmentation technique.

20.3.3 Minimum Distance Classification

The minimum distance classifier is another simple way to model the clusters. Each cluster is simply represented by its center of mass $\boldsymbol{m}_q$. Based on this model, a simple partition of the feature space is given by searching for the minimum distance from the feature vector to each of the classes. To perform this operation, we simply compute the distance of the feature vector $\boldsymbol{m}$ to each cluster center $\boldsymbol{m}_q$:

$$d_q^2 = |\boldsymbol{m} - \boldsymbol{m}_q|^2 = \sum_{p=1}^{P} (m_p - m_{qp})^2. \tag{20.5}$$

The feature is then assigned to the class to which it has the shortest distance.

Geometrically, this approach partitions the feature space as illustrated in Fig. 20.10. The boundaries between the clusters are hyper-

planes perpendicular to the vectors connecting the cluster centers at a distance halfway between them.

The minimum distance classifier, like the box classifier, requires a number of computations that is proportional to the dimension of the feature space and the number of clusters. It is a flexible technique that can be modified in various ways.

The size of the cluster could be taken into account by introducing a scaling factor into the distance computation Eq. (20.5). In this way, a feature must be closer to a narrow cluster to be associated with it. Secondly, we can define a maximum distance for each class. If the distance of a feature is larger than the maximum distance for all clusters, the object is rejected as not belonging to any of the identified classes.

20.3.4 Maximum Likelihood Classification

The maximum likelihood classifier models the clusters as statistical probability density functions. In the simplest case, P-dimensional normal distributions are taken. Given this model, we compute for each feature vector the probability that it belongs to any of the P classes. We can then associate the feature vector with the class for which it has the maximum likelihood. The new aspect with this technique is that probabilistic decisions are possible. It is not required that we decide to put an object into a certain class. We can simply give the object probabilities for membership in the different classes.

20.4 Further Readings‡

From the vast amount of literature about classification, we mention only three monographs here. The book of Schürmann [166] shows in a unique the common concepts of classification techniques based on classical statistical techniques and on neural networks. The application of neural networks for classification is detailed by Bishop [9]. One of the most recent advances in classification, the so-called *support vector machines*, are very readably introduced by Christianini and Shawe-Taylor [20].

Part V

Reference Part

A Reference Material

Selection of CCD imaging sensors (Section 1.7.1)

R1

(C: saturation capacity in 1000 electrons [ke], eNIR: enhanced NIR sensitivity, FR: frame rate in s^{-1}, ID: image diagonal in mm, QE: peak quantum efficiency)

Chip	Format H × V	FR	ID	Pixel size H × V, μm	Comments
Interlaced EIA video					
Sony[1] ICX258AL	768 × 494	30	6.09	6.35 × 7.4	1/3", eNIR
Sony[1] ICX248AL	768 × 494	30	8.07	8.4 × 9.8	1/2", eNIR
Sony[1] ICX082AL	768 × 494	30	11.1	11.6 × 13.5	2/3"
Interlaced CCIR video					
Sony[1] ICX259AL	752 × 582	25	6.09	6.5 × 6.25	1/3", eNIR
Sony[1] ICX249AL	752 × 582	25	8.07	8.6 × 8.3	1/2", eNIR
Sony[1] ICX083AL	752 × 582	25	10.9	11.6 × 11.2	2/3"
Progressive scanning interline					
Sony[1] ICX098AL	659 × 494	30	4.61	5.6 × 5.6	1/4"
Sony[1] ICX084AL	659 × 494	30	6.09	7.4 × 7.4	1/3"
Sony[1] ICX204AL	1024 × 768	15	5.95	4.65 × 4.65	1/3"
Kodak[2] KAI-0311M	648 × 484	30	7.28	9.0 × 9.0	1/2", QE 0.37 @ 500 nm
Sony[1] ICX074AL	659 × 494	40	8.15	9.9 × 9.9	1/2", C 32 ke, QE 0.43 @ 340 nm
Sony[1] ICX075AL	782 × 582	30	8.09	8.3 × 8.3	1/2"
Sony[1] ICX205AL	1360 × 1024	9.5	7.72	4.65 × 4.65	1/2", C 12 ke
Sony[1] ICX285AL	1360 × 1024	10	11.0	6.45 × 6.45	2/3", eNIR, C 18 ke, QE 0.65 @ 500 nm
Sony[1] ICX085AL	1300 × 1030	12.5	11.1	6.7 × 6.7	2/3", C 20 ke, QE 0.54 @ 380 nm
Kodak[2] KAI-1020M	1000 × 1000	49	10.5	7.4 × 7.4	QE 0.45 @ 490 nm
Kodak[2] KAI-1010M	1008 × 1018	30	12.9	9.0 × 9.0	QE 0.37 @ 500 nm
Kodak[2] KAI-2000M	1600 × 1200	30	14.8	7.4 × 7.4	QE 0.36 @ 490 nm
Kodak[2] KAI-4000M	2048 × 2048	15	21.4	7.4 × 7.4	QE 0.36 @ 490 nm

[1] http://www.sony.co.jp/en/Products/SC-HP/Product_List_E/index.html

[2] http://www.kodak.com/go/ccd

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2

Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

R2 **Selection of CMOS imaging sensors (Section 1.7.1)**

(C: saturation capacity in 1000 electrons [ke], FR: frame rate in s^{-1}, PC: pixel clock in MHz, QE: peak quantum efficiency)

Chip	Format H × V	FR	PC	Pixel size H × V, μm	Comments
Linear response					
PhotonFocus[1]	640 × 480	30	10	10.5 × 10.5	32% fill factor
Kodak[2] KAC-0311	640 × 480	60	20	7.8 × 7.8	C 45 ke, QE 0.22 @ 500 nm
Kodak[2] KAC-1310	1280 × 1024	15	20	6.0 × 6.0	C 40 ke
Fast frame rate linear response					
Fillfactory[3] LUPA1300	1280 × 1024	450	40	12.0 × 12.0	16 parallel ports
Photobit[4] PB-MV40	2352 × 1728	240	80	7.0 × 7.0	16 parallel 10-bit ports
Photobit[4] PB-MV13	1280 × 1024	600	80	12.0 × 12.0	10 parallel 10-bit ports
Photobit[4] PB-MV02	512 × 512	4000	80	16.0 × 16.0	16 parallel 10-bit ports
Logarithmic response					
IMS HDRC VGA [5]	640 × 480	25	8	12 × 12	
PhotonFocus[1]	1024 × 1024	28	28	10.6 × 10.6	linear response at low light levels with adjustable transition to logarithmic response

[1] http://www.photonfocus.com
[2] http://www.kodak.com/go/ccd
[3] http://www.fillfactory.com
[4] http://www.photobit.com
[5] http://www.ims-chips.de

Imaging sensors for the infrared (IR, Section 1.7.1)

R3

(C: full well capacity in millions electrons [Me], IT: integration time, NETD: Rausch"aquivalente Temperaturdifferenz, QE: peak quantum efficiency)

Chip	Format H × V	FR	PC	Pixel size H × V, μm	Comments
Near infrared (NIR)					
Indigo[1] InGaAs	320 × 256	345		30 × 30	0.9–1.68 μm, C 3.5 Me
Mid wave infrared (MWIR)					
AIM[2] PtSi	640 × 486	50	12	24 × 24	3.0–5.0 μm, NETD < 75 mK @ 33 ms IT
Indigo[1] InSb	320 × 256	345		30 × 30	2.0–5.0 μm, C 18 Me
Indigo[1] InSb	640 × 512	100		25 × 25	2.0–5.0 μm, C 11 Me
AIM[2] HgCdTe	384 × 288	120	20	24 × 24	3.0–5.0 μm, NETD < 20 mK @ 2 ms IT
AIM[2]/IaF FhG[3] QWIP	640 × 512	30	18	24 × 24	3.0–5.0 μm, NETD < 15 mK @ 20 ms IT
Long wave infrared (LWIR)					
AIM[2] HgCdTe	256 × 256	200	16	40 × 40	8–10 μm, NETD < 20 mK @ 0.35 ms IT
Indigo[1] QWIP	320 × 256	345		30 × 30	8.0–9.2 μm, C 18 Me, NETD < 30 mK
AIM[2]/IaF FhG[3] QWIP	256 × 256	200	16	40 × 40	8.0–9.2 μm, NETD < 8 mK @ 20 ms IT
AIM[2]/IaF FhG[3] QWIP	640 × 512	30	18	24 × 24	8.0–9.2 μm, NETD < 10 mK @ 30 ms IT
Uncooled sensors					
Indigo[1] Microbolometer	320 × 240	60		30 × 30	7.0–14.0 μm, NETD < 120 mK

[1] http://www.indogosystems.com

[2] http://www.aim-ir.de

[3] http://www.iaf.fhg.de/ir/qwip/index.html

R4 **Properties of the W-dimensional Fourier transform (Section 2.3.5)**

$g(\boldsymbol{x}) \circ\!\!-\!\!\bullet\, \hat{g}(\boldsymbol{k})$ and $h(\boldsymbol{x}) \circ\!\!-\!\!\bullet\, \hat{h}(\boldsymbol{k})$ are Fourier transform pairs: $\mathbb{R}^W \mapsto \mathbb{C}$:

$$\hat{g}(\boldsymbol{k}) = \int_{-\infty}^{\infty} g(\boldsymbol{x}) \exp\left(-2\pi \mathrm{i} \boldsymbol{k}^T \boldsymbol{x}\right) \mathrm{d}^W x = \left\langle \exp\left(2\pi \mathrm{i} \boldsymbol{k}^T \boldsymbol{x}\right) | g(\boldsymbol{x}) \right\rangle$$

s is a real, nonzero number, a and b are complex constants; $\boldsymbol{A}$ and $\boldsymbol{U}$ are W×W matrices, $\boldsymbol{R}$ is an orthogonal rotation matrix ($\boldsymbol{R}^{-1} = \boldsymbol{R}^T$, $\det R = 1$)

Property	Spatial domain	Fourier domain
Linearity	$ag(\boldsymbol{x}) + bh(\boldsymbol{x})$	$a\hat{g}(\boldsymbol{k}) + b\hat{h}(\boldsymbol{k})$
Similarity	$g(s\boldsymbol{x})$	$\hat{g}(\boldsymbol{k}/s)/\lvert s\rvert$
Generalized similarity	$g(\boldsymbol{A}\boldsymbol{x})$	$\hat{g}\left((\boldsymbol{A}^{-1})^T \boldsymbol{k}\right) / \det \boldsymbol{A}$
Rotation	$g(\boldsymbol{R}\boldsymbol{x})$	$\hat{g}(\boldsymbol{R}\boldsymbol{k})$
Separability	$\prod_{w=1}^{W} g_w(x_w)$	$\prod_{w=1}^{W} \hat{g}_w(k_w)$
Shift in x space	$g(\boldsymbol{x} - \boldsymbol{x}_0)$	$\exp(-2\pi \mathrm{i} \boldsymbol{k}^T \boldsymbol{x}_0)\hat{g}(\boldsymbol{k})$
Finite difference	$g(\boldsymbol{x} + \boldsymbol{x}_0/2) - g(\boldsymbol{x} - \boldsymbol{x}_0/2)$	$2\mathrm{i} \sin(\pi \boldsymbol{x}_0^T \boldsymbol{k})\hat{g}(\boldsymbol{k})$
Shift in k space	$\exp(2\pi \mathrm{i} \boldsymbol{k}_0^T \boldsymbol{x})g(\boldsymbol{x})$	$\hat{g}(\boldsymbol{k} - \boldsymbol{k}_0)$
Modulation	$\cos(2\pi \boldsymbol{k}_0^T \boldsymbol{x})g(\boldsymbol{x})$	$(\hat{g}(\boldsymbol{k} - \boldsymbol{k}_0) + \hat{g}(\boldsymbol{k} + \boldsymbol{k}_0))/2$
Differentiation in x space	$\dfrac{\partial g(\boldsymbol{x})}{\partial x_p}$	$2\pi \mathrm{i} k_p \hat{g}(\boldsymbol{k})$
Differentiation in k space	$-2\pi \mathrm{i} x_p g(\boldsymbol{x})$	$\dfrac{\partial \hat{g}(\boldsymbol{k})}{\partial k_p}$
Definite integral, mean	$\int_{-\infty}^{\infty} g(\boldsymbol{x}') \mathrm{d}^W x'$	$\hat{g}(\boldsymbol{0})$
Moments	$\int_{-\infty}^{\infty} x_p^m x_q^n g(\boldsymbol{x}) \mathrm{d}^W x$	$\left(\dfrac{\mathrm{i}}{2\pi}\right)^{m+n} \left(\dfrac{\partial^{m+n} \hat{g}(\boldsymbol{k})}{\partial k_p^m \partial k_q^n}\right)\Big\rvert_{\boldsymbol{0}}$
Convolution	$\int_{-\infty}^{\infty} h(\boldsymbol{x}') g(\boldsymbol{x} - \boldsymbol{x}') \mathrm{d}^W x'$	$\hat{h}(\boldsymbol{k})\hat{g}(\boldsymbol{k})$
Spatial correlation	$\int_{-\infty}^{\infty} h(\boldsymbol{x}') g(\boldsymbol{x}' + \boldsymbol{x}) \mathrm{d}^W x'$	$\hat{g}^*(\boldsymbol{k})\, \hat{h}(\boldsymbol{k})$
Multiplication	$h(\boldsymbol{x})g(\boldsymbol{x})$	$\int_{-\infty}^{\infty} \hat{h}(\boldsymbol{k}') \hat{g}(\boldsymbol{k} - \boldsymbol{k}') \mathrm{d}^W k'$
Inner product	$\int_{-\infty}^{\infty} g^*(\boldsymbol{x})\, h(\boldsymbol{x}) \mathrm{d}^W x$	$\int_{-\infty}^{\infty} \hat{g}^*(\boldsymbol{k}) \hat{h}(\boldsymbol{k}) \mathrm{d}^W k$

Elementary transform pairs for the continuous Fourier transform

R5

2-D and 3-D functions are marked by † and ‡, respectively.

Space domain	Fourier domain
Delta, $\delta(x)$	const., 1
const., 1	Delta, $\delta(k)$
$\cos(k_0 x)$	$\frac{1}{2}(\delta(k-k_0)+\delta(k+k_0))$
$\sin(k_0 x)$	$\frac{\mathrm{i}}{2}(\delta(k-k_0)-\delta(k+k_0))$
$\operatorname{sgn}(x)=\begin{cases}1 & x\geq 0\\ -1 & x<0\end{cases}$	$\frac{-\mathrm{i}}{\pi k}$
Box, $\Pi(x)=\begin{cases}1 & \lvert x\rvert<1/2\\ 0 & \lvert x\rvert\geq 1/2\end{cases}$	$\operatorname{sinc}(k)=\frac{\sin(\pi k)}{\pi k}$
Disk, † $\frac{1}{\pi r^2}\Pi\left(\frac{\lvert\boldsymbol{x}\rvert}{2r}\right)$	Bessel, $\frac{J_1(2\pi r\lvert\boldsymbol{k}\rvert)}{\pi r\lvert\boldsymbol{k}\rvert}$
Ball, ‡ $\Pi\left(\frac{\lvert\boldsymbol{x}\rvert}{2}\right)$	$\frac{\sin(\lvert\boldsymbol{k}\rvert)-\lvert\boldsymbol{k}\rvert\cos(\lvert\boldsymbol{k}\rvert)}{\lvert\boldsymbol{k}\rvert^3/(4\pi)}$
Bessel, $\frac{J_1(2\pi x)}{x}$	$2(1-k)^{1/2}\Pi\left(\frac{k}{2}\right)$
$\exp(-\lvert x\rvert)$, $\exp(-\lvert\boldsymbol{x}\rvert)^\dagger$	$\frac{2}{1+(2\pi k)^2}$, $\frac{2\pi}{(1+(2\pi\lvert\boldsymbol{k}\rvert)^2)^{3/2}}$ †

Functions invariant under the Fourier transform

R6

Space domain	Fourier domain
Gaussian, $\exp\left(-\pi\boldsymbol{x}^T\boldsymbol{x}\right)$	Gaussian, $\exp\left(-\pi\boldsymbol{k}^T\boldsymbol{k}\right)$
$x_p\exp\left(-\pi\boldsymbol{x}^T\boldsymbol{x}\right)$	$-\mathrm{i}k_p\exp\left(-\pi\boldsymbol{k}^T\boldsymbol{k}\right)$
$\operatorname{sech}(\pi x)=\frac{1}{\exp(\pi x)+\exp(-\pi x)}$	$\operatorname{sech}(\pi k)=\frac{1}{\exp(\pi k)+\exp(-\pi k)}$
Hyperbola, $\lvert\boldsymbol{x}\rvert^{-W/2}$	$\lvert\boldsymbol{k}\rvert^{-W/2}$
1-D δ comb, $\text{Ш}(x)=\sum_{n=-\infty}^{\infty}\delta(x-n)$	$\text{Ш}(k)=\sum_{v=-\infty}^{\infty}\delta(k-v)$

R7 **Properties of the 2-D DFT (Section 2.3.5)**

$\boldsymbol{G}$ and $\boldsymbol{H}$ are complex-valued M×N matrices, $\hat{\boldsymbol{G}}$ and $\hat{\boldsymbol{H}}$ their Fourier transforms,

$$\begin{aligned}\hat{g}_{u,v} &= \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} g_{m,n} \mathrm{w}_M^{-mu}\mathrm{w}_N^{-nv},\ \mathrm{w}_N = \exp(2\pi \mathrm{i}/N)\\ g_{m,n} &= \sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\hat{g}_{u,v}\mathrm{w}_M^{mu}\mathrm{w}_N^{nv},\end{aligned}$$

and a and b complex-valued constants. Stretching and replication by factors $K, L \in \mathbb{N}$ yields KM×LN matrices. For proofs see Cooley and Tukey [22], Poularikas [141].

Property	Space domain	Wave-number domain
Mean	$\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} G_{mn}$	$\hat{g}_{0,0}$
Linearity	$a\boldsymbol{G} + b\boldsymbol{H}$	$a\hat{\boldsymbol{G}} + b\hat{\boldsymbol{H}}$
Spatial stretching (up-sampling)	$g_{Km,Ln}$	$\hat{g}_{uv}/(KL)$ $(\hat{g}_{kM+u,lN+v} = \hat{g}_{u,v})$
Replication (frequency stretching)	$g_{m,n}$ $(g_{kM+m,lN+n} = g_{m,n})$	$\hat{g}_{Ku,Lv}$
Shifting	$g_{m-m',n-n'}$	$\mathrm{w}_M^{-m'u}\mathrm{w}_N^{-n'v}\hat{g}_{uv}$
Modulation	$\mathrm{w}_M^{u'm}\mathrm{w}_N^{v'n}g_{m,n}$	$\hat{g}_{u-u',v-v'}$
Finite differences	$(g_{m+1,n} - g_{m-1,n})/2$ $(g_{m,n+1} - g_{m,n-1})/2$	$\mathrm{i}\sin(2\pi u/M)\hat{g}_{uv}$ $\mathrm{i}\sin(2\pi v/N)\hat{g}_{uv}$
Convolution	$\sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1} h_{m'n'}g_{m-m',n-n'}$	$MN\hat{h}_{uv}\hat{g}_{uv}$
Spatial correlation	$\sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1} h_{m'n'}g_{m+m',n+n'}$	$MN\hat{h}_{uv}\hat{g}^*_{uv}$
Multiplication	$g_{mn}h_{mn}$	$\sum_{u'=0}^{M-1}\sum_{v'=0}^{N-1} h_{u'v'}g_{u-u',v-v'}$
Inner product	$\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} g^*_{mn}h_{mn}$	$\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\hat{g}^*_{uv}\hat{h}_{uv}$
Norm	$\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \lvert g_{mn}\rvert^2$	$\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\lvert\hat{g}_{uv}\rvert^2$

Properties of the continuous 1-D Hartley transform (Section 2.4.2) R8

$g(x) \circ\!\!-\!\!\bullet\, \hat{g}(k)$ and $h(x) \circ\!\!-\!\!\bullet\, \hat{h}(k)$ are Hartley transform pairs: $\mathbb{R} \mapsto \mathbb{R}$,

$$ {}^{h}\hat{g}(k) = \int_{-\infty}^{\infty} g(x)\,\mathrm{cas}(2\pi kx)\mathrm{d}x \quad \circ\!\!-\!\!\bullet \quad g(x) = \int_{-\infty}^{\infty} {}^{h}\hat{g}(k)\,\mathrm{cas}(2\pi kx)\mathrm{d}k $$

with

$$ \mathrm{cas}\,2\pi kx = \cos(2\pi kx) + \sin(2\pi kx). $$

s is a real, nonzero number, a and b are real constants.

Property	Spatial domain	Fourier domain
Linearity	$ag(x) + bh(x)$	$a\hat{g}(k) + b\hat{h}(k)$
Similarity	$g(sx)$	$\hat{g}(k/s)/\lvert s\rvert$
Shift in x space	$g(x - x_0)$	$\cos(2\pi kx_0)\hat{g}(k) - \sin(2\pi kx_0)\hat{g}(-k)$
Modulation	$\cos(2\pi k_0 x)g(x)$	$(\hat{g}(k - k_0) + \hat{g}(k + k_0))/2$
Differentiation in x space	$\dfrac{\partial g(\boldsymbol{x})}{\partial x_p}$	$-2\pi k_p \hat{g}(-k)$
Definite integral, mean	$\displaystyle\int_{-\infty}^{\infty} g(\boldsymbol{x}')\mathrm{d}x'$	$\hat{g}(0)$
Convolution	$\displaystyle\int_{-\infty}^{\infty} h(x')g(x - x')\mathrm{d}x'$	$[\hat{g}(k)\hat{h}(k) + \hat{g}(k)\hat{h}(-k) + \hat{g}(-k)\hat{h}(k) - \hat{g}(-k)\hat{h}(-k)]/2$
Multiplication	$h(x)g(x)$	$[\hat{g}(k) * \hat{h}(k) + \hat{g}(k) * \hat{h}(-k) + \hat{g}(-k) * \hat{h}(k) - \hat{g}(-k) * \hat{h}(-k)]/2$
Autocorrelation	$\displaystyle\int_{-\infty}^{\infty} g(x')g(x' + x)\mathrm{d}x'$	$[\hat{g}^2(k) + \hat{g}^2(-k)]/2$

1. Fourier transform expressed in terms of the Hartley transform

$$ \hat{g}(k) = \frac{1}{2}\left({}^{h}\hat{g}(k) + {}^{h}\hat{g}(-k)\right) - \frac{\mathrm{i}}{2}\left({}^{h}\hat{g}(k) - {}^{h}\hat{g}(-k)\right) $$

2. Hartley transform expressed in terms of the Fourier transform

$$ {}^{h}\hat{g}(k) = \Re[\hat{g}(k)] - \Im[\hat{g}(k)] = \frac{1}{2}\left(\hat{g}(k) + \hat{g}^*(k)\right) + \frac{\mathrm{i}}{2}\left(\hat{g}(k) - \hat{g}^*(k)\right) $$

R9 **Probability density functions (PDFs, Section 3.4).**

Definition, mean, and variance of some PDFs

Name	Definition	Mean	Variance
Discrete PDFs f_n			
Poisson $P(\mu)$	$\exp(-\mu)\dfrac{\mu^n}{n!},\ n \geq 0$	μ	μ
Binomial $B(Q,p)$	$\dfrac{Q!}{n!\,(Q-n)!}p^n(1-p)^{Q-n},\ 0 \leq n < Q$	Qp	$Qp(1-p)$
Continuous PDFs $f(x)$			
Uniform $U(a,b)$	$\dfrac{1}{b-a}$	$\dfrac{a+b}{2}$	$\dfrac{(b-a)^2}{12}$
Normal $N(\mu,\sigma)$	$\dfrac{1}{\sqrt{2\pi}\sigma}\exp\left(-\dfrac{(x-\mu)^2}{2\sigma^2}\right)$	μ	σ^2
Rayleigh $R(\sigma)$	$\dfrac{x}{\sigma^2}\exp\left(-\dfrac{x^2}{2\sigma^2}\right),\ x>0$	$\sigma\sqrt{\pi/2}$	$\sigma^2(4-\pi)/2$
Chi-square $\chi^2(Q,\sigma)$	$\dfrac{x^{Q/2-1}}{2^{Q/2}\Gamma(Q/2)\sigma^Q}\exp\left(-\dfrac{x}{2\sigma^2}\right),\ x>0$	$Q\,\sigma^2$	$2Q\,\sigma^4$

Addition theorems for independent random variables g_1 and g_2

PDF	g_1	g_2	g_1+g_2
Binomial	$B(Q_1,p)$	$B(Q_2,p)$	$B(Q_1+Q_2,p)$
Poisson	$P(\mu_1)$	$P(\mu_2)$	$P(\mu_1+\mu_2)$
Normal	$N(\mu_1,\sigma_1)$	$N(\mu_2,\sigma_2)$	$N(\mu_1+\mu_2,(\sigma_1^2+\sigma_2^2)^{1/2})$
Chi-square	$\chi^2(Q_1,\sigma)$	$\chi^2(Q_2,\sigma)$	$\chi^2(Q_1+Q_2,\sigma)$

PDFs of functions of independent random variables g_n

PDF of variable	Function	PDF of function
g_n: $N(0,\sigma)$	$(g_1^2+g_2^2)^{1/2}$	$R(\sigma)$
g_n: $N(0,\sigma)$	$\arctan(g_2^2/g_1^2)$	$U(0,2\pi)$
g_n: $N(0,\sigma)$	$\displaystyle\sum_{n=1}^{Q} g_n^2$	$\chi^2(Q,\sigma)$

Error propagation (Sections 3.2.3, 3.3.3, and 4.2.8) **R10**

f_g is the PDF of the random variable (RV) g, a, and b constants, $g' = p(g)$ a differentiable monotonic function with the derivative $\mathrm{d}p/\mathrm{d}g$ and the inverse function $g = p^{-1}(g')$.
Let $\boldsymbol{g}$ be a vector with P RVs with the covariance matrix $\mathrm{cov}(\boldsymbol{g})$, $\boldsymbol{g}'$ a vector with Q RVs and with the covariance matrix $\mathrm{cov}(\boldsymbol{g}')$, $\boldsymbol{M}$ a $Q \times P$ matrix, and $\boldsymbol{a}$ a column vector with Q elements.

1. PDF, mean, and variance of a linear function $g' = ag + b$

$$f_{g'}(g') = \frac{f_g((g'-a)/b)}{|a|}, \quad \mu_{g'} = a\mu_g + b, \quad \sigma_{g'}^2 = a^2\sigma_g^2$$

2. PDF of monotonous differentiable nonlinear function $g' = p(g)$

$$f_{g'}(g') = \frac{f_g(p^{-1}(g'))}{|\mathrm{d}p(p^{-1}(g'))/\mathrm{d}g|},$$

3. Mean and variance of differentiable nonlinear function $g' = p(g)$

$$\mu_{g'} \approx p(\mu_g) + \frac{\sigma_g^2}{2}\frac{\mathrm{d}^2 p(\mu_g)}{\mathrm{d}g^2}, \quad \sigma_{g'}^2 \approx \left|\frac{\mathrm{d}p(\mu_g)}{\mathrm{d}g}\right|^2 \sigma_g^2$$

4. Covariance matrix of a linear combination of RVs, $\boldsymbol{g}' = \boldsymbol{M}\boldsymbol{g} + \boldsymbol{a}$

$$\mathrm{cov}(\boldsymbol{g}') = \boldsymbol{M}\,\mathrm{cov}(\boldsymbol{g})\boldsymbol{M}^T$$

5. Covariance matrix of a nonlinear combination of RVs, $\boldsymbol{g}' = \boldsymbol{p}(\boldsymbol{g})$

$$\mathrm{cov}(\boldsymbol{g}') \approx \boldsymbol{J}\,\mathrm{cov}(\boldsymbol{g})\boldsymbol{J}^T \quad \text{with the Jacobian } \boldsymbol{J}, \quad j_{q,p} = \frac{\partial p_q}{\partial g_p}.$$

6. Homogeneous stochastic field: convolution of a random vector by the filter $\boldsymbol{h}$ $\boldsymbol{g}' = \boldsymbol{h} * \boldsymbol{g}$ (Section 4.2.8)
   (a) With the autocovariance vector $\boldsymbol{c}$

$$\boldsymbol{c}' = \boldsymbol{c} \star (\boldsymbol{h} \star \boldsymbol{h}) \quad \circ\!\!-\!\!\bullet \quad \hat{\boldsymbol{c}}'(k) = \hat{\boldsymbol{c}}(k)\left|\hat{\boldsymbol{h}}(k)\right|^2.$$

   (b) With the autocovariance vector $\boldsymbol{c} = \sigma^2\delta_n$ (uncorrelated elements)

$$\boldsymbol{c}' = \sigma^2(\boldsymbol{h} \star \boldsymbol{h}) \quad \circ\!\!-\!\!\bullet \quad \hat{\boldsymbol{c}}'(k) = \sigma^2\left|\hat{\boldsymbol{h}}(k)\right|^2.$$

R11 **1-D LSI filters (Sections 4.2.6, 11.2, and 12.2)**

1. Transfer function of a 1-D filter with an odd number of coefficients ($2R+1$, $[h_{-R},\ldots,h_{-1},h_0,h_1,\ldots,h_R]$)
   (a) General
   $$\hat{h}(\tilde{k}) = \sum_{v'=-R}^{R} h_{v'} \exp(-\pi \mathrm{i} v' \tilde{k})$$
   (b) Even symmetry ($h_{-v} = h_v$)
   $$\hat{h}_v = h_0 + 2\sum_{v'=1}^{R} h_{v'} \cos(\pi v' \tilde{k})$$
   (c) Odd symmetry ($h_{-v} = -h_v$)
   $$\hat{h}_v = -2\mathrm{i}\sum_{v'=1}^{R} h_{v'} \sin(\pi v' \tilde{k})$$
2. Transfer function of a 1-D filter with an even number of coefficients ($2R$, $[h_{-R},\ldots,h_{-1},h_1,\ldots,h_R]$, convolution results put on intermediate grid)
   (a) Even symmetry ($h_{-v} = h_v$)
   $$\hat{h}_v = 2\sum_{v'=1}^{R} h_{v'} \cos(\pi (v'-1/2) \tilde{k})$$
   (b) Odd symmetry ($h_{-v} = -h_v$)
   $$\hat{h}_v = -2\mathrm{i}\sum_{v'=1}^{R} h_{v'} \sin(\pi (v'-1/2) \tilde{k})$$

1-D recursive filters (Section 4.3). **R12**

1. General filter equation

$$g'_n = -\sum_{n''=1}^{S} a_{n''} g'_{n-n''} + \sum_{n'=-R}^{R} h_{n'} g_{n-n'}$$

2. General transfer function

$$\hat{h}(\tilde{k}) = \frac{\sum_{n'=-R}^{R} h_{n'} \exp(-\pi i n' \tilde{k})}{\sum_{n''=0}^{S} a_{n''} \exp(-\pi i n'' \tilde{k})}$$

3. Factorization of the transfer function using the z transform and the fundamental law of algebra

$$\hat{h}(z) = h_{-R} z^{S-R} \frac{\prod_{n'=1}^{2R} (1 - c_{n'} z^{-1})}{\prod_{n''=1}^{S} (1 - d_{n''} z^{-1})}$$

4. Relaxation filter
   (a) Filter equation ($|\alpha| < 1$)

$$g'_n = \alpha g'_{n\mp 1} + (1 - \alpha) g_n$$

   (b) Point spread function

$$^{\pm}r_{\pm n} = \begin{cases} (1-\alpha)\alpha^n & n \geq 0 \\ 0 & \text{else} \end{cases}$$

   (c) Transfer function of symmetric filter (running filter successively in positive and negative direction)

$$\hat{r}(\tilde{k}) = \frac{1}{1 + \beta - \beta \cos \pi \tilde{k}}, \quad \left(\hat{r}(0) = 1, \hat{r}(1) = \frac{1}{1 + 2\beta} \right)$$

   with

$$\beta = \frac{2\alpha}{(1-\alpha)^2}, \quad \alpha = \frac{1 + \beta - \sqrt{1 + 2\beta}}{\beta}, \quad \beta \in]-1/2, \infty[$$

5. Resonance filter with unit response at resonance wave number $\tilde{k}_0$ in the limit of low damping $1-r \ll 1$

   (a) Filter equation (damping coefficient $r \in [0,1[$, resonance wave number $\tilde{k}_0 \in [0,1]$)

$$g'_n = (1-r^2)\sin(\pi\tilde{k}_0)g_n + 2r\cos(\pi\tilde{k}_0)g'_{n\mp1} - r^2 g'_{n\mp2}$$

   (b) Point spread function

$$h_{\pm n} = \begin{cases} (1-r^2)r^n \sin[(n+1)\pi\tilde{k}_0] & n \geq 0 \\ 0 & n < 0 \end{cases}$$

   (c) Transfer function of symmetric filter (running filter successively in positive and negative direction)

$$\hat{s}(\tilde{k}) = \frac{\sin^2(\pi\tilde{k}_0)(1-r^2)^2}{\left(1-2r\cos[\pi(\tilde{k}-\tilde{k}_0)]+r^2\right)\left(1-2r\cos[\pi(\tilde{k}+\tilde{k}_0)]+r^2\right)}$$

   (d) For low damping, the transfer function can be approximated by

$$\hat{s}(\tilde{k}) \approx \frac{1}{1+(\tilde{k}-\tilde{k}_0)^2 \Big/ \frac{(1-r^2)^2}{4r^2\pi^2}} \quad \text{for} \quad 1-r \ll 1$$

   (e) Halfwidth Δk, defined by $\hat{s}(\tilde{k}_0+\Delta k) = 1/2$

$$\Delta k \approx (1-r)/\pi$$

R13 Gaussian and Laplacian pyramids (Section 5.3)

1. Construction of the *Gaussian pyramid* $\boldsymbol{G}^{(0)}, \boldsymbol{G}^{(1)}, \ldots, \boldsymbol{G}^{(Q-1)}$ with Q planes by iterative smoothing and subsampling by a factor of two in all directions

$$\boldsymbol{G}^{(0)} = \boldsymbol{G}, \quad \boldsymbol{G}^{(q+1)} = \mathcal{B}_{\downarrow 2}\boldsymbol{G}^{(q)}$$

2. Condition for smoothing filter to avoid aliasing

$$\hat{B}(\tilde{\boldsymbol{k}}) = 0 \quad \forall \tilde{k}_p \geq \frac{1}{2}$$

3. Construction of the *Laplacian pyramid* $\boldsymbol{L}^{(0)}, \boldsymbol{L}^{(1)}, \ldots, \boldsymbol{L}^{(Q-1)}$ with Q planes from the Gaussian pyramid

$$\boldsymbol{L}^{(q)} = \boldsymbol{G}^{(q)} - \uparrow_2 \boldsymbol{G}^{(q+1)}, \quad \boldsymbol{L}^{(Q-1)} = \boldsymbol{G}^{(Q-1)}$$

   The last plane of the Laplacian pyramid is the last plane of the Gaussian pyramid.

4. Interpolation filters for upsampling operation $\uparrow_2$ (≻ R22)
5. Iterative reconstruction of the original image from the Laplacian pyramid. Compute
$$\boldsymbol{G}^{(q-1)} = \boldsymbol{L}^{(q-1)} + \uparrow_2 \boldsymbol{G}^q$$
starting with the highest plane ($q = Q - 1$). When the same upsampling operator is used as for the construction of the Laplacian pyramid, the reconstruction is perfect except for rounding errors.
6. Directio-pyramidal decomposition in two directional components
$$\begin{aligned}
\boldsymbol{G}^{(q+1)} &= \downarrow_2 \mathcal{B}_x \mathcal{B}_y \boldsymbol{G}^{(q)} \\
\boldsymbol{L}^{(q)} &= \boldsymbol{G}^{(q)} - \uparrow_2 \boldsymbol{G}^{(q+1)} \\
\boldsymbol{L}_x^{(q)} &= 1/2(\boldsymbol{L}^{(q)} - (\mathcal{B}_x - \mathcal{B}_y)\boldsymbol{G}^{(q)}) \\
\boldsymbol{L}_y^{(q)} &= 1/2(\boldsymbol{L}^{(q)} + (\mathcal{B}_x - \mathcal{B}_y)\boldsymbol{G}^{(q)})
\end{aligned}$$

Basic properties of electromagnetic waves (Section 6.2) R14

1. The *frequency* ν (cycles per unit time) and *wavelength* λ (length of a period) are related by the *phase speed* c (in vacuum *speed of light* $c = 2.9979 \times 10^8\,\mathrm{m\,s^{-1}}$):
$$\lambda\nu = c$$
2. Classification of the ultraviolet, visible and infrared part of the electromagnetic spectrum (see also Fig. 6.2)

Name	Wavelength range	Comment
VUV (vacuum UV)	30–180 nm	Strongly absorbed by air; requires evacuated equipment
UV-C	100–280 nm	CIE standard definition
UV-B	280–315 nm	CIE standard definition
UV-A	315–400 nm	CIE standard definition
Visible (light)	400–700 nm	Visible by the human eye
VNIR (very near IR)	0.7–1.0 μm	IR wavelength range to which standard silicon image sensors respond
NIR (near IR)	0.7–3.0 μm	
TIR (thermal IR)	3.0–14.0 μm	Range of largest emission at environmental temperatures
MIR (middle IR)	3–100 μm	
FIR (far IR)	100–1000 μm	

3. Energy and momentum of particulate radiation such as β radiation (electrons), α radiation (helium nuclei), neutrons, and photons (electromagnetic radiation):

$$\begin{aligned} \nu &= E/h \quad \text{Bohr frequency condition,} \\ \lambda &= h/p \quad \text{de Broglie wavelength relation.} \end{aligned}$$

R15 Radiometric and photometric terms (Section 6.3)

dA_0 is an element of area in the surface, θ the angle of incidence, Ω the solid angle. For energy-, photon-, and photometry-related terms, often the indices e, p, and v, respectively, are used.

Term	Energy-related	Photon-related	Photometric quantity
Energy	Radiant energy Q [Ws]	Photon number [1]	Luminous energy [lm s]
Energy flux (power)	Radiant flux $\Phi = \frac{dQ}{dt}$ [W]	Photon flux [s^{-1}]	Luminous flux [lumen (lm)]
Incident energy flux density	Irradiance $E = \frac{d\Phi}{dA_0}$ [$W\,m^{-2}$]	Photon irradiance [$m^{-2}s^{-1}$]	Illuminance [lm/m^2 = lux [(lx)]
Excitant energy flux density	Radiant excitance (emittance) $M = \frac{d\Phi}{dA_0}$ [$W\,m^{-2}$]	Photon flux density [$m^{-2}s^{-1}$]	Luminous excitance [lm/m^2]
Energy flux per solid angle	Radiant intensity $I = \frac{d\Phi}{d\Omega}$ [Wsr^{-1}]	Photon intensity [$s^{-1}sr^{-1}$]	Luminous intensity [lm/sr = candela (cd)]
Energy flux density per solid angle	Radiance $L = \frac{d^2\Phi}{d\Omega dA_0 \cos\theta}$ [$W\,m^{-2}\ sr^{-1}$]	Photon radiance [$m^{-2}s^{-1}sr^{-1}$]	Luminance [$cd\,m^{-2}$]
Energy/area	Energy density [$W\,s\,m^2$]	Photon density [m^{-2}]	Exposure [$lm\,s\,m^{-2}$ = lx s]

Computation of luminous quantities from the corresponding radiometric quantity by the *spectral luminous efficacy* $V(\lambda)$ for daylight (photopic) vision:

$$Q_v = 683\frac{\text{lm}}{\text{W}}\int_{380\,\text{nm}}^{780\,\text{nm}} Q(\lambda)V(\lambda)\,\text{d}\lambda$$

Table with the 1980 CIE values of the spectral luminous efficacy $V(\lambda)$ for photopic vision

λ [μm]	$V(\lambda)$	λ [μm]	$V(\lambda)$	λ [μm]	$V(\lambda)$
380	0.00004	520	0.710	660	0.061
390	0.00012	530	0.862	670	0.032
400	0.0004	540	0.954	680	0.017
410	0.0012	550	0.995	690	0.0082
420	0.0040	560	0.995	700	0.0041
430	0.0116	570	0.952	710	0.0021
440	0.023	580	0.870	720	0.00105
450	0.038	590	0.757	730	0.00052
460	0.060	600	0.631	740	0.00025
470	0.091	610	0.503	750	0.00012
480	0.139	620	0.381	760	0.00006
490	0.208	630	0.265	770	0.00003
500	0.323	640	0.175	780	0.000015
510	0.503	650	0.107		

Color systems (Section 6.3.4) **R16**

1. Human color vision based on three types of cones with maximal sensitivities at 445 nm, 535 nm, and 575 nm (Fig. 6.5b).
2. *RGB* color system; additive color system with the three primary colors red, green, and blue. This could either be monochromatic colors with wavelengths 700 nm, 646.1 nm, and 435.8 nm or red, green, and blue phosphor as used in *RGB* monitors (e. g., according to the European EBU norm). Not all colors can be represented by the *RGB* color system (see Fig. 6.6a).
3. Chromaticity diagram: reduction of the 3-D color space to a 2-D color plane normalized by the intensity:

$$r = \frac{R}{R+G+B}, \quad g = \frac{G}{R+G+B}, \quad b = \frac{B}{R+G+B}.$$

   It is sufficient to use the two components r and g: $b = 1 - r - g$.
4. *XYZ* color system (Fig. 6.6c): additive color system with three virtual primaries X, Y, and Z that can represent all possible colors and is given by the following linear transform from the EBU *RGB* color

system:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.490 & 0.310 & 0.200 \\ 0.177 & 0.812 & 0.011 \\ 0.000 & 0.010 & 0.990 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

5. Color difference or YUV system: color system with an origin at the white point (Fig. 6.6b).
6. Hue-saturation (HSI) color system: color system using polar coordinates in a color difference system. The saturation is given by the radius and the hue by the angle.

R17 Thermal emission (Section 6.4.1)

1. Spectral emittance (law of Planck)

$$M_e(\lambda, T) = \frac{2\pi hc^2}{\lambda^5} \frac{1}{\exp\left(\frac{hc}{k_B T \lambda}\right) - 1}$$

with

$h = 6.6262 \times 10^{-34}\,\mathrm{J\,s}$	Planck constant,
$k_B = 1.3806 \times 10^{-23}\,\mathrm{J\,K^{-1}}$	Boltzmann constant, and
$c = 2.9979 \times 10^{8}\,\mathrm{m\,s^{-1}}$	speed of light in vacuum.

2. Total emittance (law of Stefan and Boltzmann)

$$M_e = \frac{2}{15} \frac{k_B^4 \pi^5}{c^2 h^3} T^4 = \sigma T^4 \quad \text{with} \quad \sigma \approx 5.67 \cdot 10^{-8} \mathrm{W\,m^{-2}K^{-4}}$$

3. Wavelength of maximum emittance (Wien's law)

$$\lambda_m \approx \frac{2898\mathrm{K}\,\mu\mathrm{m}}{T}$$

R18 Interaction of radiation with matter (Section 6.4)

1. *Snell's law* of *refraction* at the boundary of two optical media with the indices of refraction n_1 and n_2

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{n_2}{n_1}$$

θ_1 and θ_2 are the angles of incidence and refraction, respectively.

2. *Reflectivity* ρ: ratio of the reflected radiant flux to the incident flux at the surface. *Fresnel's equations* give the reflectivity for parallel polarized light

$$\rho_{\parallel} = \frac{\tan^2(\theta_1 - \theta_2)}{\tan^2(\theta_1 + \theta_2)},$$

for perpendicular polarized light

$$\rho_\perp = \frac{\sin^2(\theta_1 - \theta_2)}{\sin^2(\theta_1 + \theta_2)},$$

and for unpolarized light

$$\rho = \frac{\rho_\parallel + \rho_\perp}{2}.$$

3. Reflectivity at normal incidence ($\theta_1 = 0$) for all polarization states

$$\rho = \frac{(n_1 - n_2)^2}{(n_1 + n_2)^2} = \frac{(n-1)^2}{(n+1)^2} \quad \text{with} \quad n = n_1/n_2$$

4. *Total reflection.* When a ray enters into a medium with lower refractive index, beyond the critical angle θ_c all light is reflected and none enters the optically thinner medium:

$$\theta_c = \arcsin \frac{n_1}{n_2} \quad \text{with} \quad n_1 < n_2$$

Optical imaging

R19

1. Perspective projection with *pinhole camera* model

$$x_1 = -\frac{d' X_1}{X_3}, \quad x_2 = -\frac{d' X_2}{X_3}$$

Pinhole located at origin of world coordinate system $[X_1, X_2, X_3]^T$, d' is distance of image plane to projection center, X_3 axis aligned perpendicular to image plane.

2. Image equation (Newtonian and Gaussian form)

$$dd' = f^2 \quad \text{or} \quad \frac{1}{d' + f} + \frac{1}{d + f} = \frac{1}{f}$$

d and d' are the distances of the object and image to the front and back focal points of the optical system, respectively (see Fig. 7.7).

3. Lateral magnification

$$m_l = \frac{x_1}{X_1} = \frac{f}{d} = \frac{d'}{f}$$

4. Axial magnification

$$m_a \approx \frac{d'}{d} = \frac{f^2}{d^2} = \frac{d'^2}{f^2} = m_l^2$$

5. The f-number n_f of an optical system is the ratio of the focal length and diameter of lens aperture

$$n_f = \frac{f}{2r}$$

6. Depth of focus (image space)

$$\Delta x_3 = 2n_f \left(1 + \frac{d'}{f}\right) \epsilon = 2n_f(1 + m_l)\epsilon$$

7. Depth of field (object space)

Distant objects ($\Delta X_3 \ll d$)	ΔX_3	$\approx$	$2n_f \cdot \frac{1 + m_l}{m_l^2}\epsilon$
$d_{\min}$ for range including infinity	$d_{\min}$	$\approx$	$\frac{f^2}{4n_f\epsilon}$
Microscopy ($m_l \gg 1$)	ΔX_3	$\approx$	$\frac{2n_f\epsilon}{m_l}$

8. Resolution with a diffraction-limited optical systems: angular resolution

Angular resolution	$\Delta\theta_0$	$=$	$0.61\frac{\lambda}{r}$
Lateral resolution at image plane	Δx	$=$	$0.61\frac{\lambda}{n'_a}$
Lateral resolution at object plane	ΔX	$=$	$0.61\frac{\lambda}{n_a}$

The resolution is given by the Rayleigh criterion (see Fig. 7.15b); n_a and $n_{a'}$ are the object-sided and image sided numerical aperture of the light cone entering the optical system:

$$n_a = n \sin\theta_0 = \frac{2n}{n_f} = \frac{nr}{f};$$

n is the index of refraction.

9. Relation of the irradiance at image plane E' to the object radiance L (see Fig. 7.10)

$$E' = t\pi \left(\frac{r}{f + d'}\right)^2 \cos^4\theta\, L \approx t\pi \frac{\cos^4\theta}{n_f^2} L \quad \text{for} \quad d \gg f$$

Homogeneous point operation (Section 10.2) R20

Point operation that is independent of the position of the pixel

$$G'_{mn} = P(G_{mn})$$

1. Negative
$$P_N(q) = Q - 1 - q$$
2. Detection of underflow and overflow by a pseudocolor $[r, g, b]$ mapping
$$P_{uo}(q) = \begin{cases} [0, 0, Q-1] & \text{(blue)} \quad q = 0 \\ [q, q, q] & \text{(grey)} \quad q \in [1, Q-2] \\ [Q-1, 0, 0] & \text{(red)} \quad q = Q-1 \end{cases}$$
3. Contrast stretching of range $[q_1, q_2]$
$$P_{uo}(q) = \begin{cases} 0 & q < q_1 \\ \dfrac{(q - q_1)(Q-1)}{q_2 - q_1} & q \in [q_1, q_2] \\ Q - 1 & q > q_2 \end{cases}$$

Calibration procedures R21

1. Noise equalization (Section 10.2.3)
If the variance of the noise depends on the image intensity, it can be equalized by a nonlinear grayscale transformation
$$g' = h(g)\sigma_h \int_0^g \frac{\mathrm{d}g'}{\sqrt{\sigma^2(g')}} + C$$
with the two free parameters σ_h and C. With a linear variance function (Section 3.4.5)
$$\sigma_g^2(g) = \sigma_0^2 + \alpha g$$
the transformation becomes
$$h(g) = \frac{2\sigma_h}{\sqrt{\alpha}} \sqrt{\sigma_0^2 + \alpha g} + C.$$
2. Linear photometric two-point calibration (Section 10.3.3)
Two calibration images are taken, a dark image $\boldsymbol{B}$ without any illumination and a reference image $\boldsymbol{R}$ with an object of constant radiance. A normalized image corrected for both the fixed pattern noise and inhomogeneous sensitivity is given by
$$\boldsymbol{G}' = c \frac{\boldsymbol{G} - \boldsymbol{B}}{\boldsymbol{R} - \boldsymbol{B}}.$$

R22 Interpolation (Section 10.6)

1. Interpolation of continuous function from sampled points at distances Δx_w is an convolution operation:

$$g_r(\boldsymbol{x}) = \sum_{\boldsymbol{n}} g(\boldsymbol{x_n}) h(\boldsymbol{x} - \boldsymbol{x_n}).$$

   Reproduction of the grid points results in the *interpolation condition*

$$h(\boldsymbol{x_n}) = \begin{cases} 1 & \boldsymbol{n} = \boldsymbol{0} \\ 0 & \text{otherwise} \end{cases}.$$

2. Ideal interpolation function

$$h(\boldsymbol{x}) = \prod_{w=1}^{W} \operatorname{sinc}(x_w/\Delta x_w) \quad \circ\!\!-\!\!\bullet \quad \hat{h}(\boldsymbol{k}) = \prod_{w=1}^{W} \Pi(\tilde{k}_w/2)$$

3. Discrete 1-D interpolation filters for interpolation of intermediate grid points halfway between the existing points

Type	Mask	Transfer function
Linear	$\left[\begin{array}{cc} 1 & 1 \end{array}\right]/2$	$\cos(\pi\tilde{k}/2)$
Cubic	$\left[\begin{array}{cccc} -1 & 9 & 9 & -1 \end{array}\right]/16$	$\frac{9\cos(\pi\tilde{k}/2) - \cos(3\pi\tilde{k}/2)}{8}$
Cubic B-spline	$\left[\begin{array}{ccc} 1 & 23 & 23 & 1 \end{array}\right]/48$ $\left[\begin{array}{cc} 3-\sqrt{3}, & \sqrt{3}-2 \end{array}\right]/2^{\dagger}$	$\frac{23\cos(\pi\tilde{k}/2) + \cos(3\pi\tilde{k}/2)}{16 + 8\cos(\pi\tilde{k})}$

†Recursive filter applied in forward and backward direction, see Section 10.6.5

Averaging convolution filters (Chapter 11)

1. Summary of general constraints for averaging convolution filters

Property	Space domain	Wave-number domain
Preservation of mean	$\sum_n h_n = 1$	$\hat{h}(\mathbf{0}) = 1$
Zero shift, even symmetry	$h_{-\boldsymbol{n}} = h_{\boldsymbol{n}}$	$\Im\left(\hat{h}(\boldsymbol{k})\right) = 0$
Monotonic decrease from one to zero	—	$\hat{h}(\tilde{k}_2) \leq \hat{h}(\tilde{k}_1)$ if $\tilde{k}_2 > \tilde{k}_1$, $\hat{h}(\boldsymbol{k}) \in [0,1]$
Isotropy	$h(\boldsymbol{x}) = h(\lvert\boldsymbol{x}\rvert)$	$\hat{h}(\boldsymbol{k}) = \hat{h}(\lvert\boldsymbol{k}\rvert)$

2. 1-D smoothing box filters

Mask	Transfer function	Noise suppression†
${}^3\boldsymbol{R} = [1\ 1\ 1]/3$	$\frac{1}{3} + \frac{2}{3}\cos(\pi\tilde{k})$	$\frac{1}{\sqrt{3}} \approx 0.577$
${}^4\boldsymbol{R} = [1\ 1\ 1\ 1]/4$	$\cos(\pi\tilde{k})\cos(\pi\tilde{k}/2)$	$1/2 = 0.5$
${}^{2R+1}\boldsymbol{R} = [1\ \ldots\ 1]/(2R+1)$	$\frac{\sin(\pi(2R+1)\tilde{k}/2)}{(2R+1)\sin(\pi\tilde{k}/2)}$	$\frac{1}{\sqrt{2R+1}}$
${}^{2R}\boldsymbol{R} = [1\ \ldots\ 1]/(2R)$	$\frac{\sin(\pi R\tilde{k})}{2R\sin(\pi\tilde{k}/2)}$	$\frac{1}{\sqrt{2R}}$

†For white noise

3. 1-D smoothing binomial filters

Mask	TF	Noise suppression†
$\boldsymbol{B}^2 = [1\ 2\ 1]/4$	$\cos^2(\pi\tilde{k}/2)$	$\sqrt{\frac{3}{8}} \approx 0.612$
$\boldsymbol{B}^4 = [1\ 4\ 6\ 4\ 1]/16$	$\cos^4(\pi\tilde{k}/2)$	$\sqrt{\frac{35}{128}} \approx 0.523$
$\boldsymbol{B}^{2R}$	$\cos^{2R}(\pi\tilde{k}/2)$	$\left(\frac{\Gamma(R+1/2)}{\sqrt{\pi}\,\Gamma(R+1)}\right)^{1/2} \approx \left(\frac{1}{R\pi}\right)^{1/4}\left(1 - \frac{1}{16R}\right)$

†For white noise

R24 First-order derivative convolution filters (Chapter 12)

1. Summary of general constraints for a first-order derivative filter into the direction x_w

Property	Space domain	Wave-number domain
Zero mean	$\sum_{\boldsymbol{n}} h_{\boldsymbol{n}} = 0$	$\hat{h}(\tilde{\boldsymbol{k}})\big\vert_{\tilde{k}_w=0} = 0$
Zero shift, odd symmetry	$h_{-\boldsymbol{n}} = -h_{\boldsymbol{n}}$	$\Re\left(\hat{H}(\boldsymbol{k})\right) = 0$
First-order derivative	$\sum_{\boldsymbol{n}} n_w h_{\boldsymbol{n}} = 1$	$\frac{\partial \hat{h}(\tilde{\boldsymbol{k}})}{\partial \tilde{k}_w}\Big\vert_{\tilde{k}_w=0} = \pi \mathrm{i}$
Isotropy		$\hat{h}(\tilde{\boldsymbol{k}}) = \pi \mathrm{i} \tilde{k}_w \hat{b}(\vert\tilde{\boldsymbol{k}}\vert)$ with $\hat{b}(0) = 1$, $\nabla_k \hat{b}(\vert\tilde{\boldsymbol{k}}\vert) = \boldsymbol{0}$

2. First-order discrete difference filters

Name	Mask	Transfer function
$\mathcal{D}_x$	$\begin{bmatrix} 1 & -1 \end{bmatrix}$	$2\mathrm{i}\sin(\pi\tilde{k}_x/2)$
Symmetric difference, $\mathcal{D}_{2x}$	$\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}/2$	$\mathrm{i}\sin(\pi\tilde{k}_x)$
Cubic B-spline $\mathcal{D}_{2x}{}^{\pm}\mathcal{R}$	$\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}/2$, $\begin{bmatrix} 3-\sqrt{3}, & \sqrt{3}-2 \end{bmatrix}^{\dagger}$	$\mathrm{i}\dfrac{\sin(\pi\tilde{k}_x)}{2/3+1/3\cos(\pi\tilde{k}_x)}$

†Recursive filter applied in forward and backward direction, see Section 10.6.5

3. Regularized first-order discrete difference filters

Name	Mask	Transfer function
2×2, $\mathcal{D}_x\mathcal{B}_y$	$\frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$	$2\mathrm{i}\sin(\pi\tilde{k}_x/2)\cos(\pi\tilde{k}_y/2)$
Sobel, $\mathcal{D}_{2x}\mathcal{B}_y^2$	$\frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$\mathrm{i}\sin(\pi\tilde{k}_x)\cos^2(\pi\tilde{k}_y/2)$
Optimized Sobel $\mathcal{D}_{2x}(3\mathcal{B}_y^2+\mathcal{I})/4$	$\frac{1}{32}\begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix}$	$\mathrm{i}\sin(\pi\tilde{k}_x)(3\cos^2(\pi\tilde{k}_y/2)+1)/4$

4. Performance characteristics of edge detectors: angle error, magnitude error, and noise suppression for white noise. The three values in the two error columns give the errors for a wave number range of 0–0.25, 0.25–0.5, and 0.5–0.75, respectively.

Name	Angle error [°]	Magnitude error	Noise factor
$\mathcal{D}_x$			$\sqrt{2} \approx 1.414$
$\mathcal{D}_{2x}$	1.36 4.90 12.66	0.026 0.151 0.398	$1/\sqrt{2} \approx 0.707$
$\mathcal{D}_{2x}{}^{\pm}\mathcal{R}$	0.02 0.33 2.26	0.001 0.023 0.220	$\sqrt{3\ln 3/\pi} \approx 1.024$
$\mathcal{D}_x\mathcal{B}_y$	0.67 2.27 5.10	0.013 0.079 0.221	1
$\mathcal{D}_{2x}\mathcal{B}_y^2$	0.67 2.27 5.10	0.012 0.053 0.070	$\sqrt{3}/4 \approx 0.433$
$\mathcal{D}_{2x}(3\mathcal{B}_y^2+\mathcal{I})/4$	0.15 0.32 0.72	0.003 0.005 0.047	$\sqrt{59}/16 \approx 0.480$

R25 Second-order derivative convolution filters (Chapter 12)

1. Summary of general constraints for a second-order derivative filter into the direction x_w

Property	Space domain	Wave-number domain
Zero mean	$\sum_{\boldsymbol{n}} h_{\boldsymbol{n}} = 0$	$\hat{h}(\tilde{\boldsymbol{k}})\big\vert_{\tilde{k}_w=0} = 0$
Zero slope	$\sum_{\boldsymbol{n}} n_w h_{\boldsymbol{n}} = 0$	$\frac{\partial \hat{h}(\tilde{\boldsymbol{k}})}{\partial \tilde{k}_w}\Big\vert_{\tilde{k}_w=0} = 0$
Zero shift, even symmetry	$h_{-\boldsymbol{n}} = h_{\boldsymbol{n}}$	$\Re\left(\hat{H}(\boldsymbol{k})\right) = 0$
Second-order derivative	$\sum_{\boldsymbol{n}} n_w^2 h_{\boldsymbol{n}} = 2$	$\frac{\partial^2 \hat{h}(\tilde{\boldsymbol{k}})}{\partial \tilde{k}_w^2}\Big\vert_{\tilde{k}_w=0} = -2\pi^2$
Isotropy		$\hat{h}(\tilde{\boldsymbol{k}}) = -(\pi \tilde{k}_w)^2 \hat{b}(\vert\tilde{\boldsymbol{k}}\vert)$ with $\hat{b}(0) = 1$, $\nabla_k \hat{b}(\vert\tilde{\boldsymbol{k}}\vert) = \boldsymbol{0}$

2. Second-order discrete difference filters

Name	Mask	Transfer function
1-D Laplace $\mathcal{D}_x^2$	$\begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$	$-4\sin^2(\pi \tilde{k}_x/2)$
2-D Laplace $\mathcal{L}$	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	$-4\sin^2(\pi \tilde{k}_x/2) - 4\sin^2(\pi \tilde{k}_y/2)$
2-D Laplace $\mathcal{L}'$	$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$4\cos^2(\pi \tilde{k}_x/2)\cos^2(\pi \tilde{k}_y/2) - 4$

B Notation

Because of the multidisciplinary nature of digital image processing, a consistent and generally accepted terminology — as in other areas — does not exist. Two basic problems must be addressed.

- *Conflicting terminology.* Different communities use different symbols (and even names) for the same terms.
- *Ambiguous symbols.* Because of the many terms used in image processing and the areas it is related to, one and the same symbol is used for multiple terms.

There exists no trivial solution to this awkward situation. Otherwise it would be available. Thus conflicting arguments must be balanced. In this textbook, the following guidelines are used:

- *Stick to common standards.* As a first guide, the symbols recommended by international organizations (such as the International Organization for Standardization, ISO) were consulted and several major reference works were compared [40, 111, 116, 141]. Additionally cross checks were made with several standard textbooks from different areas [11, 54, 133, 143]. Only in a few conflicting situations deviations from commonly accepted symbols are used.
- *Use most compact notation.* When there was a choice of different notations, the most compact and comprehensive notation was used. In rare cases, it appeared useful to use more than one notation for the same term. It is, for example, sometimes more convenient to use indexed vector components ($\boldsymbol{x} = [x_1, x_2]^T$), and sometimes to use $\boldsymbol{x} = [x, y]^T$.
- *Allow ambiguous symbols.* One and the same symbol can have different meanings. This is not so bad as it appears at first glance because from the context the meaning of the symbol becomes unambiguous. Thus care was taken that ambiguous symbols were only used when they can clearly be distinguished by the context.

In order to familiarize readers coming from different backgrounds to the notation used in this textbook, we will give here some comments on deviating notations.

B. Jähne, Digital Image Processing
ISBN 3-540-67754-2

Copyright © 2002 by Springer-Verlag
All rights of reproduction in any form reserved.

Wave number. Unfortunately, different definitions for the term *wave number* exist:

$$k' = \frac{2\pi}{\lambda} \quad \text{and} \quad k = \frac{1}{\lambda}. \tag{B.1}$$

Physicists usually include the factor 2π in the definition of the wave number: $k' = 2\pi/\lambda$, by analogy to the definition of the circular frequency $\omega = 2\pi/T = 2\pi\nu$. In optics and spectroscopy, however, it is defined as the inverse of the wavelength without the factor 2π (i. e., number of wavelengths per unit length) and denoted by $\tilde{\nu} = \lambda^{-1}$.

Imaginary unit. The imaginary unit is denoted here by i. In electrical engineering and related areas, the symbol j is commonly used.

Time series, image matrices. The standard notation for *time series* [133], $x[n]$, is too cumbersome to be used with multidimensional signals: $g[k][m][n]$. Therefore the more compact notation x_n and $g_{k,m,n}$ is chosen.

Partial derivatives. In cases were it does not lead to confusion, partial derivates are abbreviated by indexing: $\partial g/\partial x = \partial_x g = g_x$

Typeface	Description
e, i, d, w	Upright symbols have a special meaning; examples: e for the base of natural logarithm, $\mathrm{i} = \sqrt{-1}$, symbol for derivatives: $\mathrm{d}g$, $\mathrm{w} = \mathrm{e}^{2\pi\mathrm{i}}$
a, b, …	Italic (not bold): *scalar*
$\boldsymbol{g}$, $\boldsymbol{k}$, $\boldsymbol{u}$, $\boldsymbol{x}$, …	Lowercase italic bold: *vector*, i. e., a coordinate vector, a time series, row of an image, …
$\boldsymbol{G}$, $\boldsymbol{H}$, $\boldsymbol{J}$, …	Uppercase italic bold: *matrix*, *tensor*, i. e., a discrete image, a 2-D convolution mask, a structure tensor; also used for signals with more than two dimensions
$\mathcal{B}$, $\mathcal{R}$, $\mathcal{F}$, …	Caligraphic letters indicate a representation-independent *operator*
$\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{C}$	Blackboard bold letters denote sets of numbers or other quantities

Accents	Description
$\bar{\boldsymbol{k}}$, $\bar{\boldsymbol{n}}$, …	A bar indicates a *unit vector*
$\tilde{k}$, $\tilde{\boldsymbol{k}}$, $\tilde{\boldsymbol{x}}$, …	A tilde indicates a *dimensionless normalized* quantity (of a quantity with a dimension)
$\hat{\boldsymbol{G}}$, $\hat{g}(k)$, …	A hat indicates a quantity in the *Fourier domain*

Subscript	Description
g_n	Element n of the vector $\boldsymbol{g}$
g_{mn}	Element m,n of the matrix $\boldsymbol{G}$
g_p	Compact notation for first-order partial derivative of the continuous function g into the direction p: $\partial g(\boldsymbol{x})/\partial x_p$
g_{pq}	Compact notation for second-order partial derivative of the continuous function $g(\boldsymbol{x})$ into the directions p and q: $\partial^2 g(\boldsymbol{x})/(\partial x_p \partial x_q)$

Superscript	Description
$\boldsymbol{A}^{-1}$, $\boldsymbol{A}^{-g}$	Inverse of a square matrix $\boldsymbol{A}$; generalized inverse of a (non-square) matrix $\boldsymbol{A}$
$\boldsymbol{A}^T$	Transpose of a matrix
$a^\star$	Conjugate complex
$\boldsymbol{A}^\star$	Conjugate complex and transpose of a matrix

Indexing	Description
K, L, M, N	Extension of discrete images in t, z, y, and x directions
k, l, m, n	Indices of discrete images in t, z, y, and x directions
r, s, u, v	Indices of discrete images in Fourier domain in t, z, y, and x directions
P	Number of components in a multichannel image; dimension of a feature space
Q	Number of quantization levels or number of object classes
R	Size of masks for neighborhood operators
W	Dimension of an image or feature space
p, q, w	Indices of a component in a multichannel image, dimension in an image, quantization level or feature

Function	Description
$\cos(x)$	Cosine function
$\exp(x)$	Exponential function
$\mathrm{ld}(x)$	Logarithmic function to base 2
$\ln(x)$	Logarithmic function to base **e**
$\log(x)$	Logarithmic function to base 10
$\sin(x)$	Sine function
$\mathrm{sinc}(x)$	Sinc function: $\mathrm{sinc}(x) = \sin(\pi x)/(\pi x)$
$\det(\boldsymbol{G})$	Determinant of a square matrix
$\mathrm{diag}(\boldsymbol{G})$	Vector with diagonal elements of a square matrix
$\mathrm{trace}(\boldsymbol{G})$	Trace of a square matrix
$\mathrm{cov}(\boldsymbol{g})$	Covariance matrix of a random vector
$E(\mathcal{g})$, $\mathrm{var}(\boldsymbol{G})$	Expectation (mean value) and variance

Image operators	Description
$\cdot$	Pointwise multiplication of two images
$*$	Convolution
$\star$	Correlation
$\ominus, \oplus$	Morphological erosion and dilation operators
$\circ, \bullet$	Morphological opening and closing operators
$\otimes$	Morphological hit-miss operator
$\vee, \wedge$	Boolean *or* and *and* operators
$\cup, \cap$	Union and intersection of sets
$\subset, \subseteq$	Set is subset, subset or equal
$\circlearrowleft$	Shift operator
$\downarrow_s$	Sample or reduction operator: take only every sth pixel, row, etc.
$\uparrow_s$	Expansion or interpolation operator: increase resolution in every coordinate direction by a factor of s, the new points are interpolated from the available points

Symbol	Definition, [Units]	Meaning
Greek symbols		
α	$[\mathrm{m}^{-1}]$	Absorption coefficient
β	$[\mathrm{m}^{-1}]$	Scattering coefficient
$\delta(x)$, δ_n		Continuous, discrete δ distribution
Δ	$\sum_{w=1}^{W} \frac{\partial^2}{\partial x_w^2}$	Laplacian operator
ϵ	[1]	Specific emissivity
ϵ	[m]	Radius of blur disk
κ	$[\mathrm{m}^{-1}]$	Extinction coefficient, sum of absorption and scattering coefficient
$\boldsymbol{\nabla}$	$\left[\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_W}\right]^T$	Gradient operator
λ	[m]	Wavelength
ν	$[\mathrm{s}^{-1}]$, [Hz] (hertz)	Frequency
$\boldsymbol{\nabla}\times$		Rotation operator
η	$n + \mathrm{i}\xi$, [1]	Complex index of refraction
η	[1]	Quantum efficiency
ϕ	[rad], [°]	Phase shift, phase difference
ϕ_e	[rad], [°]	Azimuth angle
Φ	[J/s], [W], $[\mathrm{s}^{-1}]$, [lm]	Radiant or luminous flux
Φ_e, Φ_p	[W], $[\mathrm{s}^{-1}]$, [lm]	Energy-based radiant, photon-based radiant, and luminous flux
ρ, $\rho_\parallel$, $\rho_\perp$	[1]	Reflectivity for unpolarized, parallel polarized, and perpendicularly polarized light
ρ	$[\mathrm{kg/m}^3]$	Density
σ_x		Standard deviation of the random variable x
σ	$5.6696 \cdot 10^{-8}\mathrm{Wm}^{-2}\mathrm{K}^{-4}$	Stefan-Boltzmann constant
σ_s	$[\mathrm{m}^2]$	Scattering cross-section
τ	[1]	Optical depth (thickness)
τ	[1]	Transmissivity
τ	[s]	Time constant
θ	[rad], [°]	Angle of incidence
θ_b	[rad], [°]	Brewster angle (polarizing angle)
θ_c	[rad], [°]	Critical angle (for total reflection)
θ_e	[rad], [°]	Polar angle
θ_i	[rad], [°]	Angle of incidence

continued on next page

Symbol	Definition, [Units]	Meaning
continued from previous page		
Ω	[sr] (steradian)	Solid angle
ω	$\omega = 2\pi\nu$, [s^{-1}], [Hz]	Circular frequency
Roman symbols		
A	[m^2]	Area
$a, \boldsymbol{a}$	$\boldsymbol{a} = \boldsymbol{x}_{tt} = \boldsymbol{u}_t$, [$m/s^2$]	Acceleration
$\hat{b}(\tilde{\boldsymbol{k}})$		Transfer function of binomial mask
$\boldsymbol{B}$	[Vs/m^2]	Magnetic field
$\boldsymbol{B}$		Binomial filter mask
$\mathcal{B}$		Binomial convolution operator
c	$2.9979 \cdot 10^8\,ms^{-1}$	speed of light
$\mathbb{C}$		set of complex numbers
d	[m]	Diameter (aperture) of optics, distance
d'	[m]	Distance in image space
$\hat{d}(\tilde{\boldsymbol{k}})$		Transfer function of $\boldsymbol{D}$
D	[m^2/s]	Diffusion coefficient
$\boldsymbol{D}$		First-order difference filter mask
$\mathcal{D}$		First-order difference operator
e	$1.6022 \cdot 10^{-19}\,As$	Elementary electric charge
e	$2.718281\ldots$	Base for natural logarithm
E	[W/m^2], [lm/m^2], [lx]	Radiant (irradiance) or luminous (illuminance) incident energy flux density
$\boldsymbol{E}$	[V/m]	Electric field
$\tilde{\boldsymbol{e}}$	[1]	Unit eigenvector of a matrix
f, f_e	[m]	(Effective) focal length of an optical system
f_b, f_f	[m]	Back and front focal length
$\boldsymbol{f}$		Optical flow
$\boldsymbol{f}$		Feature vector
F	[N] (newton)	Force
$\boldsymbol{G}$		Image matrix
$\boldsymbol{H}$		General filter mask
h	$6.6262 \cdot 10^{-34}\,Js$	Planck's constant (action quantum)
$\hbar$	$h/(2\pi)$ [Js]	
i	$\sqrt{-1}$	Imaginary unit
I	[W/sr], [lm/sr]	Radiant or luminous intensity
I	[A]	Electric current

continued on next page

Symbol	Definition, [Units]	Meaning
continued from previous page		
$\boldsymbol{I}$		Identity matrix
$\mathcal{I}$		Identity operator
$\boldsymbol{J}$		Structure tensor, inertia tensor
k_B	$1.3806 \cdot 10^{-23}$ J/K	Boltzmann constant
k	$1/\lambda$, [m^{-1}]	Magnitude of wave number
$\boldsymbol{k}$	[m^{-1}]	Wave number (number of wavelengths per unit length)
$\tilde{k}$	$k\Delta x/\pi$	Wave number normalized to the maximum wave number that can be sampled (Nyquist wave number)
K_q	[l/mol]	Quenching constant
K_r	Φ_v/Φ_e, [lm/W]	Radiation luminous efficiency
K_s	Φ_v/P [lm/W]	Lighting system luminous efficiency
K_I	[1]	Indicator equilibrium constant
L	[W/(m^2sr)], [1/(m^2sr)], [lm/(m^2sr)], [cd/m^2]	Radiant (radiance) or luminous (luminance) flux density per solid angle
$\boldsymbol{L}$		Laplacian filter mask
$\mathcal{L}$		Laplacian operator
m	[kg]	Mass
m	[1]	Magnification of an optical system
$\boldsymbol{m}$		Feature vector
M	[W/m^2], [1/(s m^2)]	Excitant radiant energy flux density (excitance, emittance)
M_e	[W/m^2]	Energy-based excitance
M_p	[1/(s m^2)]	Photon-based excitance
$\mathbb{M}$		Feature space
n	[1]	Index of refraction
n_a	[1]	Numerical aperture of an optical system
n_f	f/d, [1]	Aperture of an optical system
$\tilde{\boldsymbol{n}}$	[1]	Unit vector normal to a surface
$\mathbb{N}$		Set of natural numbers: $\{0, 1, 2, \ldots\}$
p	[kg m/s], [W m]	Momentum
p	[N/m^2]	Pressure
pH	[1]	pH value, negative logarithm of proton concentration
Q	[Ws] (joule), [lm s] number of photons	Radiant or luminous energy
Q_s	[1]	Scattering efficiency factor

continued on next page

Symbol	Definition, [Units]	Meaning
continued from previous page		
r	[m]	Radius
$\boldsymbol{r}_{m,n}$	$\boldsymbol{r}_{m,n} = [m\Delta x, n\Delta y]^T$	Translation vector on grid
$\hat{\boldsymbol{r}}_{p,q}$	$\hat{\boldsymbol{r}}_{p,q} = [p/\Delta x, q/\Delta y]^T$	Translation vector on reciprocal grid
R	Φ/s, [A/W]	Responsivity of a radiation detector
$\boldsymbol{R}$		Box filter mask
$\mathbb{R}$		Set of real numbers
s	[A]	Sensor signal
T	[K]	Absolute temperature
t	[s]	Time
t	[1]	Transmittance
u	[m/s]	Velocity
$\boldsymbol{u}$	[m/s]	Velocity vector
U	[V]	Voltage, electric potential
V	[m^3]	Volume
$V(\lambda)$	[lm/W]	Spectral luminous efficacy for photopic human vision
$V'(\lambda)$	[lm/W]	Spectral luminous efficacy for scotopic human vision
w	$\mathrm{e}^{2\pi\mathrm{i}}$	
w_N	$\exp(2\pi\mathrm{i}/N)$	
$\boldsymbol{x}$	$[x, y]^T, [x_1, x_2]^T$	Image coordinates in the spatial domain
$\boldsymbol{X}$	$[X, Y, Z]^T, [X_1, X_2, X_3]^T$	World coordinates
$\mathbb{Z}$, $\mathbb{Z}^+$		Set of integers, positive integers

Bibliography

[1] E. H. Adelson and J. R. Bergen. Spatio-temporal energy models for the perception of motion. *J. Opt. Soc. Am. A*, 2:284–299, 1985.

[2] E. H. Adelson and J. R. Bergen. The extraction of spatio-temporal energy in human and machine vision. In *Proceedings Workshop on Motion: Representation and Analysis, May 1986, Charleston, South Carolina*, pp. 151–155. IEEE Computer Society, Washington, 1986.

[3] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1974.

[4] J. Anton. *Elementary Linear Algebra*. John Wiley & Sons, New York, 2000.

[5] G. R. Arce, N. C. Gallagher, and T. A. Nodes. Median filters: theory for one and two dimensional filters. JAI Press, Greenwich, USA, 1986.

[6] S. Beauchemin and J. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–467, 1996.

[7] L. M. Biberman, ed. *Electro Optical Imaging: System Performance and Modeling*. SPIE, Bellingham, WA, 2001.

[8] J. Bigün and G. H. Granlund. Optimal orientation detection of linear symmetry. In *Proceedings ICCV'87, London*, pp. 433–438. IEEE, Washington, DC, 1987.

[9] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon, Oxford, 1995.

[10] R. Blahut. *Fast Algorithms for Digital Signal Processing*. Addison-Wesley, Reading, MA, 1985.

[11] R. Bracewell. *The Fourier Transform and its Applications*. McGraw-Hill, New York, 2nd edn., 1986.

[12] C. Broit. *Optimal registrations of deformed images*. Diss., Univ. of Pennsylvania, USA, 1981.

[13] H. Burkhardt, ed. *Workshop on Texture Analysis*, 1998. Albert-Ludwigs-Universität, Freiburg, Institut für Informatik.

[14] H. Burkhardt and S. Siggelkow. Invariant features in pattern recognition - fundamentals and applications. In C. Kotropoulos and I. Pitas, eds., *Nonlinear Model-Based Image/Video Processing and Analysis*, pp. 269–307. John Wiley & Sons, 2001.

[15] P. J. Burt. The pyramid as a structure for efficient computation. In A. Rosenfeld, ed., *Multiresolution image processing and analysis*, vol. 12 of *Springer Series in Information Sciences*, pp. 6–35. Springer, New York, 1984.

[16] P. J. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Trans. COMM*, 31:532–540, 1983.

[17] P. J. Burt, T. H. Hong, and A. Rosenfeld. Segmentation and estimation of image region properties through cooperative hierarchical computation. *IEEE Trans. SMC*, 11:802–809, 1981.

[18] J. F. Canny. A computational approach to edge detection. *PAMI*, 8:679–698, 1986.

[19] R. Chelappa. *Digital Image Processing*. IEEE Computer Society Press, Los Alamitos, CA, 1992.

[20] N. Christianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, 2000.

[21] C. M. Close and D. K. Frederick. *Modelling and Analysis of Dynamic Systems*. Houghton Mifflin, Boston, 1978.

[22] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. of Comput.*, 19:297–301, 1965.

[23] J. Crank. *The Mathematics of Diffusion*. Oxford University Press, New York, 2nd edn., 1975.

[24] P.-E. Danielsson, Q. Lin, and Q.-Z. Ye. Efficient detection of second degree variations in 2D and 3D images. Technical Report LiTH-ISY-R-2155, Department of Electrical Engineering, Linköping University, S-58183 Linköping, Sweden, 1999.

[25] P. J. Davis. *Interpolation and Approximation*. Dover, New York, 1975.

[26] C. DeCusaris, ed. *Handbook of Applied Photometry*. Springer, New York, 1998.

[27] C. Demant, B. Streicher-Abel, and P. Waszkewitz. *Industrial Image Processing. Viusal Quality Control in Manufacturing*. Springer, Berlin, 1999. Includes CD-ROM.

[28] P. DeMarco, J. Pokorny, and V. C. Smith. Full-spectrum cone sensitivity functions for X-chromosome-linked anomalous trichromats. *J. of the Optical Society*, A9:1465–1476, 1992.

[29] J. Dengler. *Methoden und Algorithmen zur Analyse bewegter Realweltszenen im Hinblick auf ein Blindenhilfesystem*. Diss., Univ. Heidelberg, 1985.

[30] R. Deriche. Fast algorithms for low-level vision. *IEEE Trans. PAMI*, 12(1): 78–87, 1990.

[31] N. Diehl and H. Burkhardt. Planar motion estimation with a fast converging algorithm. In *Proc. 8th Int. Conf. Pattern Recognition, ICPR'86, October 27–31, 1986, Paris*, pp. 1099–1102. IEEE Computer Society, Los Alamitos, 1986.

[32] R. C. Dorf and R. H. Bishop. *Modern Control Systems*. Addison-Wesley, Menlo Park, CA, 8th edn., 1998.

[33] S. A. Drury. *Image Interpretation in Geology*. Chapman & Hall, London, 2nd edn., 1993.

[34] M. A. H. Elmore, W. C. *Physics of Waves*. Dover Publications, New York, 1985.

[35] A. Erhardt, G. Zinser, D. Komitowski, and J. Bille. Reconstructing 3D light microscopic images by digital image processing. *Applied Optics*, 24:194–200, 1985.

[36] J. F. S. Crawford. *Waves*, vol. 3 of *Berkely Physics Course*. McGraw-Hill, New York, 1965.

[37] O. Faugeras. *Three-dimensional Computer Vision. A Geometric Vewpoint*. MIT Press, Cambridge, MA, 1993.

[38] M. Felsberg and G. Sommer. A new extension of linear signal processing for estimating local properties and detecting features. In G. Sommer, N. Krüger, and C. Perwass, eds., *Mustererkennung 2000, 22. DAGM Symposium, Kiel*, Informatik aktuell, pp. 195–202. Springer, Berlin, 2000.

[39] R. Feynman. *Lectures on Physics*, vol. 2. Addison-Wesley, Reading, Mass., 1964.

[40] D. G. Fink and D. Christiansen, eds. *Electronics Engineers' Handbook*. McGraw-Hill, New York, 3rd edn., 1989.

[41] M. A. Fischler and O. Firschein, eds. *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*. Morgan Kaufmann, Los Altos, CA, 1987.

[42] D. J. Fleet. *Measurement of Image Velocity*. Diss., University of Toronto, Canada, 1990.

[43] D. J. Fleet. *Measurement of Image Velocity*. Kluwer Academic Publisher, Dordrecht, 1992.

[44] D. J. Fleet and A. D. Jepson. Hierarchical construction of orientation and velocity selective filters. *IEEE Trans. PAMI*, 11(3):315–324, 1989.

[45] D. J. Fleet and A. D. Jepson. Computation of component image velocity from local phase information. *Int. J. Comp. Vision*, 5:77–104, 1990.

[46] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics, Principles and Practice*. Addison Wesley, Reading, MA, 1990.

[47] W. Förstner. Image preprocessing for feature extraction in digital intensity, color and range images. In A. Dermanis, A. Grün, and F. Sanso, eds., *Geomatic Methods for the Analysis of Data in the Earth Sciences*, vol. 95 of *Lecture Notes in Earth Sciences*. Springer, Berlin, 2000.

[48] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Trans. PAMI*, 13:891–906, 1991.

[49] G. Gaussorgues. *Infrared Thermography*. Chapman & Hall, London, 1994.

[50] P. Geißler and B. Jähne. One-image depth-from-focus for concentration measurements. In E. P. Baltsavias, ed., *Proc. ISPRS Intercommission workshop from pixels to sequences, Zürich, March 22-24*, pp. 122–127. RISC Books, Coventry UK, 1995.

[51] J. Gelles, B. J. Schnapp, and M. P. Sheetz. Tracking kinesin driven movements with nanometre-scale precision. *Nature*, 331:450–453, 1988.

[52] F. Girosi, A. Verri, and V. Torre. Constraints for the computation of optical flow. In *Proceedings Workshop on Visual Motion, March 1989, Irvine, CA*, pp. 116–124. IEEE, Washington, 1989.

[53] H. Goldstein. *Classical Mechanics*. Addison-Wesley, Reading, MA, 1980.

[54] G. H. Golub and C. F. van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, 1989.

[55] R. C. Gonzalez and R. E. Woods. *Digital image processing*. Addison-Wesley, Reading, MA, 1992.

[56] G. H. Granlund. In search of a general picture processing operator. *Comp. Graph. Imag. Process.*, 8:155–173, 1978.

[57] G. H. Granlund and H. Knutsson. *Signal Processing for Computer Vision*. Kluwer, 1995.

[58] M. Groß. *Visual Computing*. Springer, Berlin, 1994.

[59] E. M. Haacke, R. W. Brown, M. R. Thompson, and R. Venkatesan. *Magnetic Resonance Imaging: Physical Principles and Sequence Design*. John Wiley & Sons, New York, 1999.

[60] M. Halloran. 700×9000 imaging on an integrated CCD wafer - affordably. *Advanced Imaging*, Jan.:46–48, 1996.

[61] J. G. Harris. *The coupled depth/slope approach to surface reconstruction*. Master thesis, Dept. Elec. Eng. Comput. Sci., Cambridge, Mass., 1986.

[62] J. G. Harris. A new approach to surface reconstruction: the coupled depth/slope model. In *1st Int. Conf. Comp. Vis. (ICCV), London*, pp. 277–283. IEEE Computer Society, Washington, 1987.

[63] H. Haußecker. *Messung und Simulation kleinskaliger Austauschvorgänge an der Ozeanoberfläche mittels Thermographie*. Diss., University of Heidelberg, Germany, 1995.

[64] H. Haußecker. Simultaneous estimation of optical flow and heat transport in infrared imaghe sequences. In *Proc. IEEE Workshop on Computer Vision beyond the Visible Spectrum*, pp. 85–93. IEEE Computer Society, Washington, DC, 2000.

[65] H. Haußecker and D. J. Fleet. Computing optical flow with physical models of brightness variation. *IEEE Trans. PAMI*, 23:661–673, 2001.

[66] E. Hecht. *Optics*. Addison-Wesley, Reading, MA, 1987.

[67] D. J. Heeger. Optical flow from spatiotemporal filters. *Int. J. Comp. Vis.*, 1:279–302, 1988.

[68] E. C. Hildreth. Computations underlying the measurement of visual motion. *Artificial Intelligence*, 23:309–354, 1984.

[69] G. C. Holst. *CCD Arrays, Cameras, and Displays*. SPIE, Bellingham, WA, 2nd edn., 1998.

[70] G. C. Holst. *Testing and Evaluation of Infrared Imaging Systems*. SPIE, Bellingham, WA, 2nd edn., 1998.

[71] G. C. Holst. *Common Sense Approach to Thermal Imaging*. SPIE, Bellingham, WA, 2000.

[72] G. C. Holst. *Electro-optical Imaging System Performance*. SPIE, Bellingham, WA, 2nd edn., 2000.

[73] B. K. Horn. *Robot Vision*. MIT Press, Cambridge, MA, 1986.

[74] S. Howell. *Handbook of CCD Astronomy*. Cambridge University Press, Cambridge, 2000.

[75] T. S. Huang, ed. *Two-dimensional Digital Signal Processing II: Transforms and Median Filters*, vol. 43 of *Topics in Applied Physics*. Springer, New

York, 1981.

[76] S. V. Huffel and J. Vandewalle. *The Total Least Squares Problem - Computational Aspects and Analysis*. SIAM, Philadelphia, 1991.

[77] K. Iizuka. *Engineering Optics*, vol. 35 of *Springer Series in Optical Sciences*. Springer, Berlin, 2nd edn., 1987.

[78] B. Jähne. Image sequence analysis of complex physical objects: nonlinear small scale water surface waves. In *Proceedings ICCV'87, London*, pp. 191–200. IEEE Computer Society, Washington, DC, 1987.

[79] B. Jähne. Motion determination in space-time images. In *Image Processing III, SPIE Proceeding 1135, international congress on optical science and engineering, Paris, 24-28 April 1989*, pp. 147–152, 1989.

[80] B. Jähne. *Spatio-temporal Image Processing*. Lecture Notes in Computer Science. Springer, Berlin, 1993.

[81] B. Jähne. *Handbook of Digital Image Processing for Scientific Applications*. CRC Press, Boca Raton, FL, 1997.

[82] B. Jähne and H. Haußecker, eds. *Computer Vision and Applications. A Guide for Students and Practitioners*. Academic Press, San Diego, 2000.

[83] B. Jähne, H. Haußecker, and P. Geißler, eds. *Handbook of Computer Vision and Applications. Volume I: Sensors and Imaging. Volume II: Signal Processing and Pattern Recognition. Volume III: Systems and Applications*. Academic Press, San Diego, 1999. Includes three CD-ROMs.

[84] B. Jähne, J. Klinke, and S. Waas. Imaging of short ocean wind waves: a critical theoretical review. *J. Optical Soc. Amer. A*, 11:2197–2209, 1994.

[85] B. Jähne, H. Scharr, and S. Körgel. Principles of filter design. In B. Jähne, H. Haußecker, and P. Geißler, eds., *Computer Vision and Applications, volume 2, Signal Processing and Pattern Recognition*, chapter 6, pp. 125–151. Academic Press, San Diego, 1999.

[86] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[87] R. Jain, R. Kasturi, and B. G. Schunck. *Machine Vision*. McGraw-Hill, New York, 1995.

[88] J. R. Janesick. *Scientific Charge-Coupled Devices*. SPIE, Bellingham, WA, 2001.

[89] J. T. Kajiya. The rendering equation. *Computer Graphics*, 20:143–150, 1986.

[90] M. Kass and A. Witkin. Analysing oriented patterns. *Comp. Vis. Graph. Im. Process.*, 37:362–385, 1987.

[91] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. In *Proc. 1st Int. Conf. Comp. Vis. (ICCV), London*, pp. 259–268. IEEE Computer Society, Washington, 1987.

[92] B. Y. Kasturi and R. C. Jain. *Computer Vision: Advances and Applications*. IEEE Computer Society, Los Alamitos, 1991.

[93] B. Y. Kasturi and R. C. Jain, eds. *Computer Vision: Principles*. IEEE Computer Society, Los Alamitos, 1991.

[94] J. K. Kearney, W. B. Thompson, and D. L. Boley. Optical flow estimation: an error analysis of gradient-based methods with local optimization. *IEEE*

Trans. PAMI, 9 (2):229–244, 1987.

[95] M. Kerckhove, ed. *Scale-Space and Morphology in Computer Vision*, vol. 2106 of *Lecture Notes in Computer Science*, 2001. 3rd Int. Conf. Scale-Space'01, Vancouver, Canada, Springer, Berlin.

[96] C. Kittel. *Introduction to Solid State Physics*. Wiley, New York, 1971.

[97] R. Klette, A. Koschan, and K. Schlüns. *Computer Vision. Three-Dimensional Data from Images*. Springer, New York, 1998.

[98] H. Knutsson. *Filtering and Reconstruction in Image Processing*. Diss., Linköping Univ., Sweden, 1982.

[99] H. Knutsson. Representing local structure using tensors. In *The 6th Scandinavian Conference on Image Analysis, Oulu, Finland, June 19-22, 1989*, 1989.

[100] H. E. Knutsson, R. Wilson, and G. H. Granlund. Anisotropic nonstationary image estimation and its applications: part I – restoration of noisy images. *IEEE Trans. COMM*, 31(3):388–397, 1983.

[101] J. J. Koenderink and A. J. van Doorn. Generic neighborhood operators. *IEEE Trans. PAMI*, 14(6):597–605, 1992.

[102] C. Koschnitzke, R. Mehnert, and P. Quick. *Das KMQ-Verfahren: Medienkompatible Übertragung echter Stereofarbabbildungen*. Forschungsbericht Nr. 201, Universität Hohenheim, 1983.

[103] P. Lancaster and K. Salkauskas. *Curve and Surface Fitting. An Introduction*. Academic Press, London, 1986.

[104] S. Lanser and W. Eckstein. Eine Modifikation des Deriche-Verfahrens zur Kantendetektion. In B. Radig, ed., *Mustererkennung 1991*, vol. 290 of *Informatik Fachberichte*, pp. 151–158. 13. DAGM Symposium, München, Springer, Berlin, 1991.

[105] Laurin. *The Photonics Design and Applications Handbook*. Laurin Publishing CO, Pittsfield, MA, 40th edn., 1994.

[106] D. C. Lay. *Linear Algebra and Its Applications*. Addison-Wesley, Reading, MA, 1999.

[107] R. Lenz. Linsenfehlerkorrigierte Eichung von Halbleiterkameras mit Standardobjektiven für hochgenaue 3D-Messungen in Echtzeit. In E. Paulus, ed., *Proc. 9. DAGM-Symp. Mustererkennung 1987, Informatik Fachberichte 149*, pp. 212–216. DAGM, Springer, Berlin, 1987.

[108] R. Lenz. Zur Genauigkeit der Videometrie mit CCD-Sensoren. In H. Bunke, O. Kübler, and P. Stucki, eds., *Proc. 10. DAGM-Symp. Mustererkennung 1988, Informatik Fachberichte 180*, pp. 179–189. DAGM, Springer, Berlin, 1988.

[109] M. Levine. *Vision in Man and Machine*. McGraw-Hill, New York, 1985.

[110] Z.-P. Liang and P. C. Lauterbur. *Principles of Magnetic Resonance Imaging: A Signal Processing Perspective*. SPIE, Bellingham, WA, 1999.

[111] D. R. Lide, ed. *CRC Handbook of Chemistry and Physics*. CRC, Boca Raton, FL, 76th edn., 1995.

[112] J. S. Lim. *Two-dimensional Signal and Image Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1990.

[113] T. Lindeberg. *Scale-space Theory in Computer Vision*. Kluwer Academic Publishers, Boston, 1994.

[114] M. Loose, K. Meier, and J. Schemmel. A self-calibrating single-chip CMOS camera with logarithmic response. *IEEE J. Solid-State Circuits*, 36(4), 2001.

[115] D. Lorenz. *Das Stereobild in Wissenschaft und Technik*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt, Köln, Oberpfaffenhofen, 1985.

[116] V. K. Madisetti and D. B. Williams, eds. *The Digital Signal Processing Handbook*. CRC, Boca Raton, FL, 1998.

[117] H. A. Mallot. *Computational Vision: Information Processing in Perception and Visual Behavior*. The MIT Press, Cambridge, MA, 2000.

[118] V. Markandey and B. E. Flinchbaugh. Multispectral constraints for optical flow computation. In *Proc. 3rd Int. Conf. on Computer Vision 1990 (ICCV'90), Osaka*, pp. 38–41. IEEE Computer Society, Los Alamitos, 1990.

[119] S. L. Marple Jr. *Digital Spectral Analysis with Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

[120] D. Marr. *Vision*. W. H. Freeman and Company, New York, 1982.

[121] D. Marr and E. Hildreth. Theory of edge detection. *Proc. Royal Society, London, Ser. B*, 270:187–217, 1980.

[122] E. A. Maxwell. *General Homogeneous Coordinates in Space of Three Dimensions*. University Press, Cambridge, 1951.

[123] C. Mead. *Analog VLSI and Neural Systems*. Addison-Wesley, Reading, MA, 1989.

[124] W. Menke. *Geophysical Data Analysis: Discrete Inverse Theory*, vol. 45 of *International Geophysics Series*. Academic Press, San Diego, 1989.

[125] C. D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia, 2001.

[126] A. Z. J. Mou, D. S. Rice, and W. Ding. VIS-based native video processing on UltraSPARC. In *Proc. IEEE Int. Conf. on Image Proc., ICIP'96*, pp. 153–156. IEEE, Lausanne, 1996.

[127] T. Münsterer. *Messung von Konzentrationsprofilen gelöster Gase in der wasserseitigen Grenzschicht*. Diploma thesis, University of Heidelberg, Germany, 1993.

[128] H. Nagel. Displacement vectors derived from second-order intensity variations in image sequences. *Computer Vision, Graphics, and Image Processing (GVGIP)*, 21:85–117, 1983.

[129] Y. Nakayama and Y. Tanida, eds. *Atlas of Visualization III*. CRC, Boca Raton, FL, 1997.

[130] V. S. Nalwa. *A Guided Tour of Computer Vision*. Addison-Wesley, Reading, MA, 1993.

[131] M. Nielsen, P. Johansen, O. Olsen, and J. Weickert, eds. *Scale-Space Theories in Computer Vision*, vol. 1682 of *Lecture Notes in Computer Science*, 1999. 2nd Int. Conf. Scale-Space'99, Corfu, Greece, Springer, Berlin.

[132] H. K. Nishihara. Practical real-time stereo matcher. *Optical Eng.*, 23:536–545, 1984.

[133] A. V. Oppenheim and R. W. Schafer. *Discrete-time Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[134] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, New York, 3rd edn., 1991.

[135] J. R. Parker. *Algorithms for Image Processing and Computer Vision*. John Wiley & Sons, New York, 1997. Includes CD-ROM.

[136] P. Perona and J. Malik. Scale space and edge detection using anisotropic diffusion. In *Proc. IEEE comp. soc. workshop on computer vision (Miami Beach, Nov. 30-Dec. 2, 1987)*, pp. 16–20. IEEE Computer Society, Washington, 1987.

[137] Photobit. PB-MV13 20 mm CMOS Active Pixel Digital Image Sensor. Photobit, Pasadena, CA, August 2000. www.photobit.com.

[138] M. Pietikäinen and A. Rosenfeld. Image segmentation by texture using pyramid node linking. *SMC*, 11:822–825, 1981.

[139] I. Pitas. *Digital Image Processing Algorithms*. Prentice Hall, New York, 1993.

[140] I. Pitas and A. N. Venetsanopoulos. *Nonlinear Digital Filters. Principles and Applications*. Kluwer Academic Publishers, Norwell, MA, 1990.

[141] A. D. Poularikas, ed. *The Transforms and Applications Handbook*. CRC, Boca Raton, 1996.

[142] W. Pratt. *Digital image processing*. Wiley, New York, 2nd edn., 1991.

[143] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, 1992.

[144] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing. Principles, Algorithms, and Applications*. McMillan, New York, 1992.

[145] L. H. Quam. Hierarchical warp stereo. In *Proc. DARPA Image Understanding Workshop, October 1984, New Orleans, LA*, pp. 149–155, 1984.

[146] A. R. Rao. *A Taxonomy for Texture Description and Identification*. Springer, New York, 1990.

[147] A. R. Rao and B. G. Schunck. Computing oriented texture fields. In *Proceedings CVPR'89, San Diego, CA*, pp. 61–68. IEEE Computer Society, Washington, DC, 1989.

[148] T. H. Reiss. *Recognizing Planar Objects Using Invariant Image Features*, vol. 676 of *Lecture notes in computer science*. Springer, Berlin, 1993.

[149] J. A. Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, Belmont, CA, 1995.

[150] A. Richards. *Alien Vision: Exploring the Electromagnetic Spectrum with Imaging Technology*. SPIE, Bellingham, WA, 2001.

[151] J. A. Richards. *Remote Sensing Digital Image Analysis*. Springer, Berlin, 1986.

[152] J. A. Richards and X. Jia. *Remote Sensing Digital Image Analysis*. Springer, Berlin, 1999.

[153] M. J. Riedl. *Optical Design Fundamentals for Infrared Systems*. SPIE, Bellingham, 2nd edn., 2001.

[154] K. Riemer. *Analyse von Wasseroberflächenwellen im Orts-Wellenzahl-Raum*. Diss., Univ. Heidelberg, 1991.

[155] K. Riemer, T. Scholz, and B. Jähne. Bildfolgenanalyse im Orts-Wellenzahlraum. In B. Radig, ed., *Mustererkennung 1991, Proc. 13. DAGM-Symposium München, 9.-11. October 1991*, pp. 223–230. Springer, Berlin, 1991.

[156] A. Rosenfeld, ed. *Multiresolution Image Processing and Analysis*, vol. 12 of *Springer Series in Information Sciences*. Springer, New York, 1984.

[157] A. Rosenfeld and A. C. Kak. *Digital Picture Processing*, vol. I and II. Academic Press, San Diego, 2nd edn., 1982.

[158] J. C. Russ. *The Image Processing Handbook*. CRC, Boca Raton, FL, 3rd edn., 1998.

[159] H. Samet. *Applications of Spatial Data Structures: Computer Graphics, Image processing, and GIS*. Addison-Wesley, Reading, MA, 1990.

[160] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA, 1990.

[161] H. Scharr and D. Uttenweiler. 3D anisotropic diffusion filtering for enhancing noisy actin filaments. In B. Radig and S. Florczyk, eds., *Pattern Recognition, 23rd DAGM Stmposium, Munich*, vol. 2191 of *Lecture Notes in Computer Science*, pp. 69–75. Springer, Berlin, 2001.

[162] H. Scharr and J. Weickert. An anisotropic diffusion algorithm with optimized rotation invariance. In G. Sommer, N. Krüger, and C. Perwass, eds., *Mustererkennung 2000*, Informatik Aktuell, pp. 460–467. 22. DAGM Symposium, Kiel, Springer, Berlin, 2000.

[163] T. Scheuermann, G. Pfundt, P. Eyerer, and B. Jähne. Oberflächenkonturvermessung mikroskopischer Objekte durch Projektion statistischer Rauschmuster. In G. Sagerer, S. Posch, and F. Kummert, eds., *Mustererkennung 1995, Proc. 17. DAGM-Symposium, Bielefeld, 13.-15. September 1995*, pp. 319–326. DAGM, Springer, Berlin, 1995.

[164] C. Schnörr and J. Weickert. Variational image motion computations: theoretical framework, problems and perspective. In G. Sommer, N. Krüger, and C. Perwass, eds., *Mustererkennung 2000*, Informatik Aktuell, pp. 476–487. 22. DAGM Symposium, Kiel, Springer, Berlin, 2000.

[165] J. R. Schott. *Remote Sensing. The Image Chain Approach*. Oxford University Press, New York, 1997.

[166] J. Schürmann. *Pattern Classification*. John Wiley & Sons, New York, 1996.

[167] R. Sedgewick. *Algorithms in C, Part 1–4*. Addison-Wesley, Reading, MA, 3rd edn., 1997.

[168] J. Serra. *Image analysis and mathematical morphology*. Academic Press, London, 1982.

[169] J. Serra and P. Soille, eds. *Mathematical Morphology and its Applications to Image Processing*, vol. 2 of *Computational Imaging and Vision*. Kluwer, Dordrecht, 1994.

[170] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger. Shiftable multiscale transforms. *IEEE Trans. IT*, 38(2):587–607, 1992.

[171] R. M. Simonds. Reduction of large convolutional kernels into multipass applications of small generating kernels. *J. Opt. Soc. Am. A*, 5:1023–1029, 1988.

[172] A. Singh. *Optic Flow Computation: a Unified Perspective*. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[173] A. T. Smith and R. J. Snowden, eds. *Visual Detection of Motion*. Academic Press, London, 1994.

[174] W. J. Smith. *Modern Optical Design*. McGraw-Hill, New York, 3rd edn., 2000.

[175] P. Soille. *Morphological Image Analysis. Principles and Applications*. Springer, Berlin, 1999.

[176] G. Sommer, ed. *Geometric Computing with Clifford Algebras*. Springer, Berlin, 2001.

[177] J. Steurer, H. Giebel, and W. Altner. Ein lichtmikroskopisches Verfahren zur zweieinhalbdimensionalen Auswertung von Oberflächen. In G. Hartmann, ed., *Proc. 8. DAGM-Symp. Mustererkennung 1986, Informatik-Fachberichte 125*, pp. 66–70. DAGM, Springer, Berlin, 1986.

[178] R. H. Stewart. *Methods of Satellite Oceanography*. University of California Press, Berkeley, 1985.

[179] T. M. Strat. Recovering the camera parameters from a transformation matrix. In *Proc. DARPA Image Understanding Workshop*, pp. 264–271, 1984.

[180] B. ter Haar Romeny, L. Florack, J. Koenderink, and M. Viergever, eds. *Scale-Space Theory in Computer Vision*, vol. 1252 of *Lecture Notes in Computer Science*, 1997. 1st Int. Conf., Scale-Space'97, Utrecht, The Netherlands, Springer, Berlin.

[181] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Trans. PAMI*, 8:413–424, 1986.

[182] D. Terzopoulos. The computation of visible-surface representations. *IEEE Trans. PAMI*, 10 (4):417–438, 1988.

[183] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-seeking models for 3D object reconstruction. In *Proc. 1st Int. Conf. Comp. Vis. (ICCV), London*, pp. 269–276. IEEE, IEEE Computer Society Press, Washington, 1987.

[184] D. H. Towne. *Wave Phenomena*. Dover, New York, 1988.

[185] S. Ullman. *High-level Vision. Object Recognition and Visual Cognition*. The MIT Press, Cambridge, MA, 1996.

[186] S. E. Umbaugh. *Computer Vision and Image Processing: A Practical Approach Using CVIPTools*. Prentice Hall PTR, Upper Saddle River, NJ, 1998.

[187] M. Unser, A. Aldroubi, and M. Eden. Fast B-spline transforms for continuous image representation and interpolation. *IEEE Trans. PAMI*, 13: 277–285, 1991.

[188] F. van der Heijden. *Image Based Measurement Systems. Object Recognition and Parameter Estimation*. Wiley, Chichester, England, 1994.

[189] W. M. Vaughan and G. Weber. Oxygen quenching of pyrenebutyric acid fluorescence in water. *Biochemistry*, 9:464, 1970.

[190] A. Verri and T. Poggio. Against quantitative optical flow. In *Proceedings ICCV'87, London*, pp. 171–180. IEEE, IEEE Computer Society Press, Washington, DC, 1987.

[191] A. Verri and T. Poggio. Motion field and optical flow: qualitative properties. *IEEE Trans. PAMI*, 11 (5):490–498, 1989.

[192] K. Voss and H. Süße. *Praktische Bildverarbeitung*. Hanser, München, 1991.

[193] B. A. Wandell. *Foundations of Vision*. Sinauer Ass., Sunderland, MA, 1995.

[194] A. Watt. *Fundamentals of Three-dimensional Computer Graphics*. Addison-Wesley, Workingham, England, 1989.

[195] J. Weickert. *Anisotropic Diffusion in Image Processing*. Dissertation, Faculty of Mathematics, University of Kaiserslautern, 1996.

[196] J. Weickert. *Anisotrope Diffusion in Image Processing*. Teubner, Stuttgart, 1998.

[197] I. Wells, W. M. Efficient synthesis of Gaussian filters by cascaded uniform filters. *IEEE Trans. PAMI*, 8(2):234–239, 1989.

[198] J. N. Wilson and G. X. Ritter. *Handbook of Computer Vision Algorithms in Image Algebra*. CRC, Boca Raton, FL, 2nd edn., 2000.

[199] G. Wiora. *Optische 3D-Messtechnik: Präzise Gestaltvermessung mit einem erweiterten Streifenprojektionsverfahren*. Dissertation, Fakultät für Physik und Astronomie, Universität Heidelberg, 2001. http://www.ub.uni-heidelberg.de/archiv/1808.

[200] G. Wolberg. *Digital Image Warping*. IEEE Computer Society, Los Alamitos, CA, 1990.

[201] R. J. Woodham. Multiple light source optical flow. In *Proc. 3rd Int. Conf. on Computer Vision 1990 (ICCV'90), Osaka*, pp. 42–46. IEEE Computer Society, Los Alamitos, 1990.

[202] P. Zamperoni. *Methoden der digitalen Bildsignalverarbeitung*. Vieweg, Braunschweig, 1989.

Index

Symbols

3-D imaging 205
4-neighborhood 33
6-neighborhood 33
8-neighborhood 33

A

absorption coefficient 170
accurate 77
acoustic imaging 153
acoustic wave 152
  longitudinal 152
  transversal 152
action quantum 150
action-perception cycle 16
active contour 442
active vision 16, 18
adder circuit 297
adiabatic compressibility 152
aerial image 514
AI 515
aliasing 233
alpha radiation 151
AltiVec 25
amplitude 56
amplitude of Fourier component 56
anaglyph method 210
analog-digital converter 247
analytic function 361
analytic signal 361
and operation 481
aperture problem 210, 379, 384, 385, 391, 394, 401, 450, 464
aperture stop 189
area 508
ARMA 116
artificial intelligence 18, 515
associativity 110, 484
astronomy 3, 18
autocorrelation function 94
autocovariance function 94
autoregressive-moving average process 116
averaging
  recursive 303
axial magnification 187

B

B-splines 276
back focal length 186
band sampling 156
band-limited 236
bandwidth-duration product 55
bandpass decomposition 135, 139
bandpass filter 121, 128
base
  orthonormal 39
basis image 39, 107
BCCE 386, 391
bed-of-nails function 236
Bessel function 199
beta radiation 151
bidirectional reflectance distribution function 170
bimodal distribution 428
binary convolution 481
binary image 36, 427
binary noise 296
binomial distribution 89, 291
binomial filter 392
bioluminescence 173
bit reversal 68, 69
blackbody 163, 166
block matching 392
Bouger's law 170
bounding box 499
box filter 286, 392
box function 196
BRDF 170
Brewster angle 169

brightness change constraint equation 386
butterfly operation 70

C

calibration error 77
camera coordinates 178
Camera link 24
Canny edge detector 333
Cartesian coordinates 90
Cartesian Fourier descriptor 504
cartography 207
Cauchy–Schwarz inequality 407
causal filter 115, 116
CCD 21
CD-ROM 24
center of mass 500
central limit theorem 90
central moment 80, 500
centroid 504
chain code 495, 498
characteristic value 114
characteristic vector 114
charge coupled device 21
chemiluminescence 173
chess board distance 34
chi density 91
chi-square density 91, 92
child node 497
circular aperture 201
circularity 510
circularly polarized 150
city block distance 34
classification 16, 516
  object-based 516
  pixel-based 516
  supervised 523
  unsupervised 523
classifier 523
closing operation 486
cluster 517
CMOS image sensor 22
co-spectrum 97
coherence 150
coherence function 97
coherency measure 349
coherency radar 8, 218
coherent 150
color difference system 161
color image 283
colorimetry 160
commutativity 109, 484
complex exponential 114, 117
complex number 41
complex plane 43
complex polynomial 117
complex-valued vector 43
computational complexity 65
computer graphics 17
computer science 17
computer vision 18
confocal laser scanning microscopy 215
connected region 32
connectivity 433
constant neighborhood 308
continuity equation 386
continuous-wave modulation 217
controlled smoothness 452
convolution 52, 86, 95, 195, 235, 350
  binary 481
  cyclic 105
  discrete 102
  normalized 309
convolution mask 52
convolution theorem 52, 108, 114
Cooley-Tukey algorithm 72
coordinates
  camera 178
  Cartesian 90
  homogeneous 183
  polar 90
  world 177
correlation 112
  cyclic 95
correlation coefficient 83
correspondence
  physical 381
  visual 381
correspondence problem 379
cosine transform 62, 63
covariance 83, 94
covariance matrix 83, 111, 465, 521
cross section 172
cross-correlation coefficient 407
cross-correlation function 95
cross-correlation spectrum 97
cross-covariance 521
cross-covariance function 95

cyclic 343
cyclic convolution 105, 297
cyclic correlation 95

D

data space 463
data vector 461, 468
decimation-in-frequency FFT 73
decimation-in-time FFT 68
decision space 523
deconvolution 113, 476
defocusing 474
deformation energy 450
degree of freedom 465
delta function, discrete 115
depth from
  multiple projections 208
  phase 207
  time-of-flight 207
  triangulation 207
depth from paradigms 207
depth imaging 205, 206
depth map 6, 213, 441
depth of field 188, 212, 477
depth of focus 187
depth range 208
depth resolution 208
depth-first traversal 497
derivation theorem 53
derivative
  directional 358
  partial 316
derivative filter 350
design matrix 461, 468
DFT 43
DHT 63
difference of Gaussian 335, 358
differential cross section 172
differential geometry 404
differential scale space 135, 139
differentiation 315
diffraction-limited optics 200
diffusion coefficient 129
diffusion equation 472
diffusion tensor 459
diffusion-reaction system 455
digital object 32
digital signal processing 77
digital video disk 24
digitization 15, 177, 233
dilation operator 482
direction 342
directional derivative 358
directiopyramidal decomposition 141, 411, 420
discrete convolution 102
discrete delta function 115
discrete difference 315
discrete Fourier transform 43, 116
discrete Hartley transform 63
discrete inverse problem 442
discrete scale space 136
disparity 209
dispersion 149
displacement vector 379, 385, 450
displacement vector field 386, 442, 450
distance transform 493
distortion
  geometric 190
distribution function 79
distributivity 110, 485
divide and conquer 65, 72
DoG 335, 358
Doppler effect 174
dual base 242
dual operators 486
duality 486
DV 379, 385
DVD 24
DVD+RW 24
DVF 386
dyadic point operator 264, 320
dynamic range 208

E

eccentricity 502
edge 308
  in tree 435
edge detection 315, 323, 339
edge detector
  regularized 331
edge strength 315
edge-based segmentation 431
effective focal length 186
effective inverse OTF 478
efficiency factor 172
eigenimage 114
eigenvalue 114, 459
eigenvalue analysis 400

eigenvalue problem 346
eigenvector 114, 399
elastic membrane 450
elastic plate 451
elastic wave 152
elasticity constant 450
electric field 147
electrical engineering 17
electromagnetic wave 147
electron 151
electron microscope 152
ellipse 502
elliptically polarized 150
emission 163
emissivity 165, 166
emittance 154
energy 58
ensemble average 93
ergodic 95
erosion operator 482
error
  calibration 77
  statistical 77
  systematic 77
error functional 446
error propagation 465
error vector 461
Ethernet 24
Euclidian distance 34
Euler-Lagrange equation 445, 455
excitance 154
expansion operator 139
expectation value 80
exponential, complex 114
exposure time 87
extinction coefficient 171

F

fan-beam projection 224
Faraday effect 173
fast Fourier transform 66
father node 435
feature 99
feature image 15, 99, 339
feature space 517
feature vector 517
FFT 66
  decimation-in-frequency 73
  decimation-in-time 68
  multidimensional 74
  radix-2 decimation-in-time 66
  radix-4 decimation-in-time 72
field
  electric 147
  magnetic 147
fill operation 500
filter 52, 99
  binomial 290
  causal 115
  difference of Gaussian 358
  finite impulse response 116
  Gabor 364, 396, 411
  infinite impulse response 116
  mask 109
  median 124, 307
  nonlinear 124
  polar separable 368
  quadrature 396
  rank value 123, 482
  recursive 115
  separable 110
  stable 116
  transfer function 109
filtered back-projection 228, 229
finite impulse response filter 116
FIR filter 116
Firewire 24
first-order statistics 78
fix point 308
fluid dynamics 386
fluorescence 173
focal plane array 533
focus series 477
forward mapping 265
four-point mapping 267
Fourier descriptor 495
  Cartesian 504
  polar 505
Fourier domain 556
Fourier ring 48
Fourier series 45, 503
Fourier slice theorem 227
Fourier torus 48
Fourier transform 29, 40, 42, 45, 95, 195
  discrete 43
  infinite discrete 45
  multidimensional 45
  one-dimensional 42
  windowed 127

Fourier transform pair 43
FPA 533
Fraunhofer diffraction 200
frequency 147
frequency doubling 149
Fresnel's equations 168
front focal length 186
FS 45

G

Gabor filter 364, 396, 411
gamma transform 253
gamma value 38
Gaussian noise 296
Gaussian probability density 89
Gaussian pyramid 126, 137, 138
generalized image coordinates 183
generalized inverse 465
geodesy 207
geometric distortion 190
geometric operation 245
geometry of imaging 177
global optimization 441
gradient space 218
gradient vector 316
gray value corner 405, 406
gray value extreme 405, 406
grid vector 34
group velocity 365

H

Haar transform 64
Hadamard transform 64
Hamilton's principle 445
Hankel transform 199
Hartley transform 63
Hesse matrix 317, 404
hierarchical processing 15
hierarchical texture organization 413
Hilbert filter 360, 411, 420
Hilbert operator 360
Hilbert space 62
Hilbert transform 359, 360
histogram 79, 517
hit-miss operator 487, 488
homogeneous 79, 107
homogeneous coordinates 183, 267
homogeneous point operation 246
homogeneous random field 94
Hough transform 437, 463
HT 63
hue 161
human visual system 18, 158
hyperplane 463

I

IA-64 25
idempotent operation 486
IDFT 45
IEEE 1394 24
IIR filter 116
illumination slicing 208
illumination, uneven 257
image analysis 427
image averaging 256
image coordinates 181
  generalized 183
image cube 381
image data compression 63
image equation 186
image flow 385
image formation 236
image preprocessing 15
image processing 17
image reconstruction 16
image restoration 16
image sensor 22
image sequence 8
image vector 467
impulse 308
impulse noise 296
impulse response 108, 115
incoherent 150
independent random variables 83
index of refraction 149
inertia tensor 356, 502
infinite discrete Fourier transform 45
infinite impulse response filter 116
infrared 23, 165
inhomogeneous background 283
inhomogeneous point operation 256
inner product 39, 42, 60, 356
input LUT 247
integrating sphere 259
intensity 161
interferometry 207
interpolation 239, 242, 269
interpolation condition 270

inverse filtering 113, 442, 476
inverse Fourier transform 42, 46
inverse mapping 265, 266
inverse problem
  overdetermined 461
irradiance 29, 154
isotropic edge detector 319
isotropy 288

J

Jacobian matrix 86, 336
joint probability density function 83
JPEG 63

K

Kerr effect 173

L

Lagrange function 445
Lambert-Beer's law 170
Lambertian radiator 164
Laplace of Gaussian 334
Laplace transform 118
Laplacian equation 449
Laplacian operator 129, 135, 317, 328
Laplacian pyramid 126, 137, 139, 411
lateral magnification 186
leaf node 497
leaf of tree 435
learning 523
least squares 447
lens aberration 474
line sampling 156
linear discrete inverse problem 461
linear interpolation 272
linear shift-invariant operator 107
linear shift-invariant system 123, 194, 474
linear symmetry 341
linear time-invariant 107
linearly polarized 149
local amplitude 362
local orientation 363, 368
local phase 362, 363
local variance 417
local wave number 358, 368, 373, 420
LoG 334
log-polar coordinates 59
logarithmic scale space 135
lognormal 369, 373
longitudinal acoustic wave 152
look-up table 247, 320
look-up table operation 247
low-level image processing 99, 427
LSI 123, 194
LSI operator 107
LTI 107
luminance 161
luminescence 173
LUT 247

M

m-rotational symmetry 505
machine vision 18
magnetic field 147
magnetic resonance 225
magnetic resonance imaging 8
magnification
  axial 187
  lateral 186
marginal probability density function 83
Marr-Hildreth operator 334
mask 100
mathematics 17
matrix 556
maximization problem 346
maximum filter 124
maximum operator 482
mean 80, 416
measurement space 517
median filter 124, 307, 314
medical imaging 18
membrane, elastic 450
memory cache 71
metameric color stimuli 159
metrology 18
MFLOP 65
microscopy 189
microwave 165
Mie scattering 172
minimum filter 124
minimum operator 482
minimum-maximum principle 133
MMX 25
model 442
model matrix 461

model space 437, 463
model vector 461
model-based segmentation 427, 436
model-based spectral sampling 156
Moiré effect 233, 237
molar absorption coefficient 171
moment 495, 500
  central 500
  scale-invariant 501
moment tensor 502
monogenic signal 363
monotony 485
morphological operator 483
motility assay 9
motion 15
motion as orientation 383
motion field 385, 386
moving average 133
MR 8, 225
multigrid representation 126, 137
Multimedia Instruction Set Extension 25
multiplier circuit 297
multiscale representation 126
multiscale texture analysis 414
multispectral image 283
multiwavelength interferometry 218

N

neighborhood
  4- 33
  6- 33
  8- 33
neighborhood operation 99
neighborhood relation 32
network model 469
neural networks 18
neutron 151
node 69
node, in tree 435
noise 283
  binary 296
  spectrum 112
  white 308
  zero-mean 94, 95
noise suppression 295, 307
non-closed boundaries 505
non-uniform illumination 283
nonlinear filter 124
nonlinear optical phenomenon 149
norm 61, 178, 461
normal density 462
normal distribution 90
normal probability density 89
normal velocity 401, 411
normalized convolution 309
null space 346
numerical aperture 202

O

object-based classification 516
occlusion 182
OCR 12, 513, 520
octree 498
OFC 386, 391
opening operation 486
operator 556
operator notation 101
operator, Laplacian 317
operator, morphological 483
optical activity 173
optical axis 178, 186
optical character recognition 12, 513, 520
optical depth 171
optical engineering 17
optical flow 385
optical flow constraint 386
optical illusions 19
optical signature 515
optical thickness 171
optical transfer function 197, 474
or operation 481
orientation 342, 343, 383, 416, 502
  local 450
orientation invariant 372
orientation vector 348
orthonormal 178
orthonormal base 39
orthonormality relation 40
OTF 197, 474, 478
outer product 46
output LUT 247
oxygen 173

P

parallax 209
parameter vector 461, 468
partial derivative 316
particle physics 3

particulate radiation 151
Pascal's triangle 292
pattern recognition 18, 513
PBA 174
PDF 79
pel 29
perimeter 509
periodicity 47, 48
  DFT 47
perspective projection 181, 182, 184
phase 56, 358, 410
phase angle 41
phase of Fourier component 56
phosphorescence 173
photogrammetry 3, 18
photography 3
photometric stereo 220, 441
photometry 156
photon 150
photonics 17
photopic vision 158
photorealistic 17, 388
physical correspondence 381
physics 17
pinhole camera 181
pixel 29, 78
pixel-based classification 516
pixel-based segmentation 427
Planck 163
Planck's constant 150
plane polarized 149
plate, elastic 451
point operation 78, 99, 245, 350
  homogeneous 246
  inhomogeneous 256
point operator 81
point spread function 108, 112, 115, 194, 428, 474
Poisson distribution 151
Poisson process 88
polar coordinates 90
polar Fourier descriptor 505
polar separable 311, 368
polarization
  circular 150
  elliptical 150
  linear 149
potential 450
power spectrum 57, 96, 112
precise 77
primary colors 160
principal axes 394
principal plane 186
principal point 186
principal ray 189
principal-axes transform 521
principle of superposition 107, 484
probability density function 79
process
  homogeneous 79
projection operator 226
projection theorem 227
proton 151
pseudo-color image 248, 250
pseudo-noise modulation 217
PSF 108, 194, 478
pulse modulation 217
pyramid 21
pyramid linking 433
pyrene butyric acid 174

Q

quad-spectrum 97
quadrant 497
quadratic scale space 135
quadrature filter 359, 364, 396
quadrature filter pair 420
quadtree 495, 496
quantization 35, 79, 177, 243
quantum efficiency 22, 92
quantum mechanics 62
quenching 173

R

radiant energy 153
radiant flux 153
radiant intensity 154
radiometric calibration
  nonlinear 260
  two-point 259
radiometry 153
radiometry of imaging 177
radiosity 388
radius 505
radix-2 FFT algorithm 66
radix-4 FFT algorithm 72
Radon transform 226
RAID array 24
random field 78, 93
  ergodic 95

homogeneous 94
random variable 79, 151
independent 83
uncorrelated 83
rank 346
rank-value filter 123, 307, 482
ratio imaging 220
Rayleigh criterion 201
Rayleigh density 90
Rayleigh theorem 58
reciprocal base 242
reciprocal grid 236
reciprocal lattice 241
reconstruction 16, 100, 441
rectangular grid 32, 33
recursive averaging 303
recursive filter 115, 116
reflectivity 168
refraction 167
region of support 100
region-based segmentation 432
regions 283
regularized edge detector 331
relaxation filter 118, 119
remote sensing 18
rendering equation 388
representation-independent notation 101
resonance filter 118
responsivity 157
restoration 100, 441, 447, 474
Riesz transform 363
robustness 351
root 308, 497
root of tree 435
rotation 35, 178, 184, 266
run-length code 495
RV 79

S

sample variance 91, 93
sampling 236
standard 239
sampling theorem 137, 234, 236, 237
satellite image 514
saturation 161
scalar 556
scalar product 39, 356
scale 128, 416
scale invariance 132, 133
scale invariant 501
scale mismatch 125
scale space 126, 128, 456
scaler circuit 296
scaling 34, 184, 266
scaling theorem 199
scotopic vision 158
searching 65
segmentation 15, 427, 442
edge-based 431
model-based 436
pixel-based 427
region-based 432
semi-group property 133
sensor element 78
separability
FT 51
separable filter 110, 118
shape 481
shape from refraction 221
shape from shading 9, 207, 218, 441
shearing 266
shift invariant 94, 107, 484
shift operator 107, 484
shift theorem 52, 57, 128, 506
shift-register stage 297
SIMD 25
similarity constraint 441
simple neighborhood 341
sine transform 62, 63
single instruction multiple data 25
singular value decomposition 463
skewness 80
smoothing filter 350
smoothness 448
smoothness constraint 441
snake 442
Snell's law 167
Sobel operator 351
software engineering 17
solid angle 154
son node 435
space-time image 381
spatiotemporal energy 396
spatiotemporal image 381
specific rotation 173
spectroradiometry 155
spectroscopic imaging 156
specular surface 168

speech processing 18
speech recognition 513
speed of light 147
speed of sound 152
spline 276
standard deviation 85
standard sampling 239
statistical error 77
steerable filter 310
Stefan-Boltzmann law 165
step edge 433
stereo image 441
stereo system 209
stereoscopic basis 209
Stern–Vollmer equation 174
stochastic process 78, 93
stretching 266
structure element 100, 483
structure tensor 439
subsampling 137
subtractor circuit 297
subtree 435
superposition principle 107, 484
supervised classification 523
support vector machine 528
symmetry 505
  DFT 48
system, linear shift-invariant 123
systematic error 77

T

target function 327
telecentric 5
telecentric illumination system 221
temperature distribution 165
tensor 556
terminal node 497
test image 289
text recognition 513
texture 15, 339, 413
theoretical mechanics 445
thermal emission 163
thermal imaging 257
thermography 165, 167
three-point mapping 267
TIFF 496
time series 58, 107, 556
tomography 16, 100, 208, 224, 441
total least squares 399
total reflection 169
tracing algorithm 432
transfer function 108, 109, 474
  recursive filter 117
translation 34, 178, 184, 266
translation invariance 499
translation invariant 107
transmission tomography 225
transmissivity 171
transmittance 171
transport equation 472
transversal acoustic wave 152
tree 435, 497
triangular grid 33
triangulation 207
tristimulus 160

U

ultrasonic microscopy 152
ultrasound 152
ultraviolet 23
uncertainty relation 55, 128, 139, 355
uncorrelated random variable 83
uneven illumination 257
uniform density 90
uniform distribution 82
unit circle 43
unit vector 556
unitary transform 29, 60
unsupervised classification 523
upsampling 51

V

Van Cittert iteration 479
variance 80, 83, 93, 416, 465
variance operator 213, 417
variation calculus 444
vector 556
vector space 44
vector, complex-valued 43
vectorial feature image 283
vertex, in tree 435
vignetting 192
VIS 25
visual computing 17
visual correspondence 381
visual inspection 5
visual instruction set 25
visual perception 18
volume element 32

volumetric image 6
volumetric imaging 205, 206
voxel 32, 381

W

Waldsterben 514
wave
  acoustic 152
  elastic 152
  electromagnetic 147
wave number 41, 155, 556
wavelength 41, 147, 155, 195
weighted averaging 309
white noise 97, 308
white point 161
white-light interferometry 8, 218
Wien's law 165
window 100
window function 238, 261
windowed Fourier transform 127
windowing 261
world coordinates 177

X

x-ray 23
x86-64 25
XYZ color system 161

Z

z-transform 48, 118
zero crossing 328, 453
zero-mean noise 94
zero-phase filter 118, 284